**CMPT 300**

**Assignment 4**

**myls command**

**Marks: 100 marks**

**Notes:**

1. **Failure to follow the instructions and rules may lead to failed test cases and/or a final grade of 0.**
2. You can do this assignment individually or in a team of two. If you are doing it in a group, only one submission per group is required.
3. You may submit multiple times until the deadline. Grade penalties will be imposed for late submissions (see the course outline for the details).
4. Always plan before coding.
5. For this assignment, you can use:

* the [standard C libraries](https://en.cppreference.com/w/c/header)
* POSIX APIs - POSIX is a standardized operating systems interface based on UNIX. You can find a list [here.](https://en.wikibooks.org/wiki/C_Programming/POSIX_Reference)

1. All the codes in this lab must be done using C language only. No other languages should be used.
2. Use function-level and inline comments throughout your code. We will not be specifically grading documentation. However, remember that you will not be able to comment on your code unless sufficiently documented. Take the time to document your code as you develop it properly.
3. We will carefully analyze the code submitted to look for plagiarism signs, so please do not do it! If you are unsure about what is allowed, please talk to an instructor or a TA.

# Coding Rules

* You must follow the file name as specified in the instructions.
* There should be **no memory leaks.**
* **Makefile**: Makefile provides the following functionality:
* **all**: compiles your program (this is the default behavior), producing an executable file named the same as the C file.
* **clean**: deletes the executable file and any intermediate files (.o, specifically)
* You will receive 0 if your makefile fails.
* Check your build to ensure that there are no errors.
* Visit TA's programming office hours to get help.

**Background**

The Unix/Linux ls command is used to list files and directories in the file system. For this assignment you will be implementing a version of ls which supports only a limited set of options.

Note: Mac-OS's ls is different; must develop under Linux.

By completing this assignment, you will:

1. Understand what an *inode* is and understand its role in the Unix file system.
2. Know how to use system calls to navigate the Unix file system from a user-level program.
3. Understand how files and directories are organized and stored in Unix.

**Assignment requirements**

In this assignment, you will be creating a program named **myls** (short for “*MY List”*).

Your program should support below mentioned functionalities.

When calling the program, it will take the form:  
./myls [options] [file list]

* + [options]: Optional options from the list below. May be specified in any order or grouping, such as “” (none), “-i”, “-i -l -R”, “-iRl” “-R -li”, “-lR”… The options are case sensitive.
  + [file list]: Optional space separated list of paths to files or directories to display.

**Options:** Implement the following options (you need *not* support any other options, and you need *not* support the long version of the option names):

-i: Print the index number of each file

-l: Use a long listing format

-R: List subdirectories recursively. (Make sure that recursion must not cause any infinite loop)

**Sort:** Sort the files and directories lexicographically. When printing directories recursively (-R option), do a depth-first traversal of the directories, entering sibling directories in lexicographical order.

*Hint :* Use ls -R to check your myls output. Both should have same order.

**Date Format**: When using the -l option you must print out date information in the following format:  
**mmm dd yyyy hh:mm**

For example:  
Oct 2 2021 14:52  
Sep 30 2021 00:01

Specifically:  
mmm Initial three characters of month name; first letter capitalized.  
dd Two digit day of month; pad with space (‘ ‘) for single digit day.  
yyyy Four digit year number; pad with spaces (‘ ‘) if less than 4 digits.  
hh Two digit hour (24 hour clock); pad with zero (‘0’) if less than 2 digits.  
mm Two digit minute; pad with zero (‘0’) if less than 2 digits.

**Special Paths**  
You must support the following paths with special meaning (x and y are arbitrary names):

/ Root directory (when at the front of path of the path such as /usr)

/x/y Absolute path to a file or directory

/xy/ Absolute path to a directory

x/y Relative path to a file or directory

x/y/ Relative path to a directory

~ User’s home folder (can be used such as ~ or ~/test/this/out)

. Current directory

.. Up one directory (can be used such as ../ or ../../oh/my/../really/.. )

\* Wildcard (such as \*.c) (most of this work will be done by the shell!)

**Simplifications vs built-in ls command**

* + - Do not print the “Total 123456” line at the top of the output when using the -l option.
    - All options (if any) must be specified before any files/directories (if any).
    - You do not need to support quoted strings as arguments:

$ myls "thank goodness we don’t/have to support this/"

$ myls 'it is like a "gift" not to do this!.txt'

$ myls 'seemed like a good idea.c'

**Error Handling**  
You must display a meaningful error and exit when:

* + - User specified an unsupported option such as -a. It is undefined behavior if the user specifies an argument such as “--version”. You should display a message “**Error: Unsupported Option”**
    - User specified a nonexistent file/directory.   
       The message should be **“Error : Nonexistent files or directories”**

**Testing**

* + Test all permutations of the possible options, including no options at all.
  + Test listing edge cases:
    - on a file; on an empty directory; on a very large directory
    - ls on different file types (normal, symbolic). Print the relevant data of what it is pointing to or the link itself. Check stat() and lstat().
  + Example commands (for easy reference)

$ myls

$ myls -lR

$ myls /

$ myls ~

$ myls ~/

$ myls / /usr

$ myls -R -il ~/cmpt300 ~/tmp /usr

$ myls ..

$ myls -R ../hello

**Hints**

* + Initially concentrate on getting the listing for a directory correct; later worry about recursion.
  + Focus on extracting the proper information and just printing it. Once you have that all done work on making the format match the ls command.
  + Consult the provided file **infodemo.c** and **secret\_hearder.h** to see what calls to use to get actual group and usernames.
  + Don't forget to test your code on directories with symbolic links in them.

**Restrictions**

* + You may *not* use 3rd party libraries. For example, you must write the path processing, file listing, and directory recursion code yourself. Standard c libraries are allowed.
  + You may not copy other people's code. If you are copying and adapting the code from external sources, credit the source.
  + Your code should make use of a modular design; try breaking your code into meaningful modules and reasonable functions!
  + You must implement sorting yourself (any *O(n2)* or better algorithm OK).   
    You may not copy-and-paste sorting code from online; you may use any built-in routines in either C or Linux which will sort. You can use alphasort().

**Test Cases**

- Run the executable with all the combinations of iRl (-i, -l, -R, -lR, -iR, -il, -iRl) including without passing an of the options

- Run the executable by passing a file or directory along with the (iRl) options

- Place a subdirectory (with a few files) inside your original directory and test if the iRl options are working fine for the files inside the subdirectory.

**Submission Instructions:**

For A4, the concrete required deliverables are:

* myls.c file
* Makefile

Submit a zip file.

**Grading Criteria**

Correct make files + 10

Prints entries for a single file argument +5

Prints entries for a single directory argument +5

Prints entries for a multiple file argument +5

Prints entries for a multiple directory argument +5

Prints entries for mixed files and directories +5

Handles case when no file or directory given, and no options +5

Option -R with sort +10

Option -i +10

Option -l +10

Option combinations: -Ri, -Rl, -il, -Ril +10

Handles symbolic links properly. +5

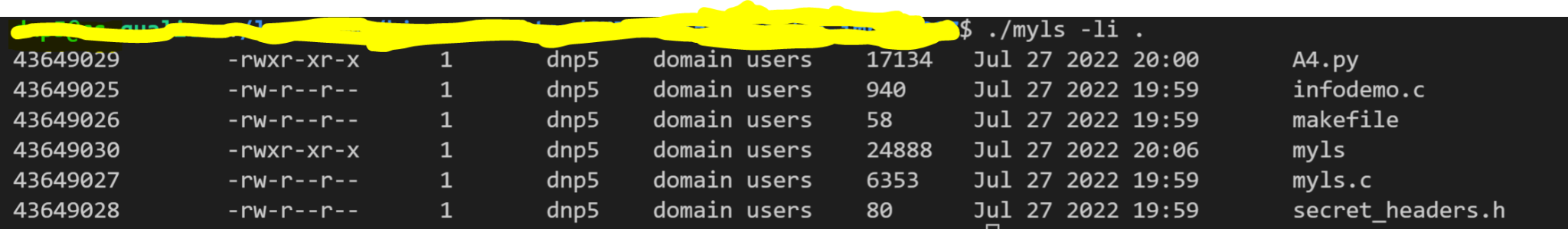
Wildcard +5

Special paths +5

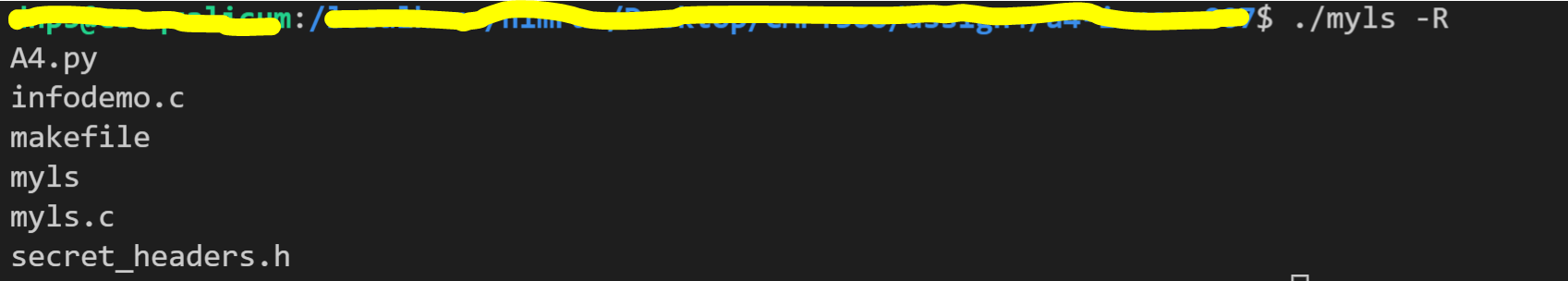
Error handling +5

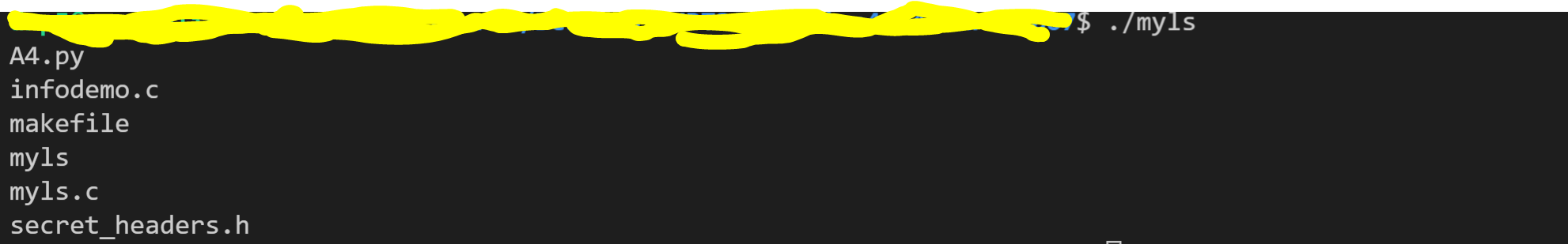
Memory leaks 50% deduction per feature

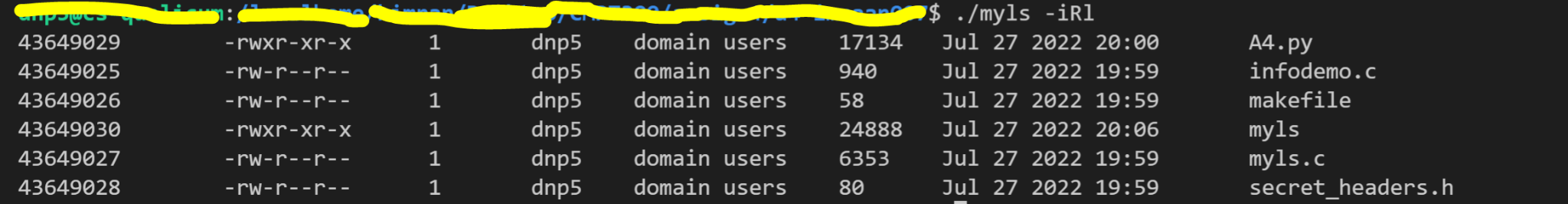
Some of the sample output screenshots for reference:

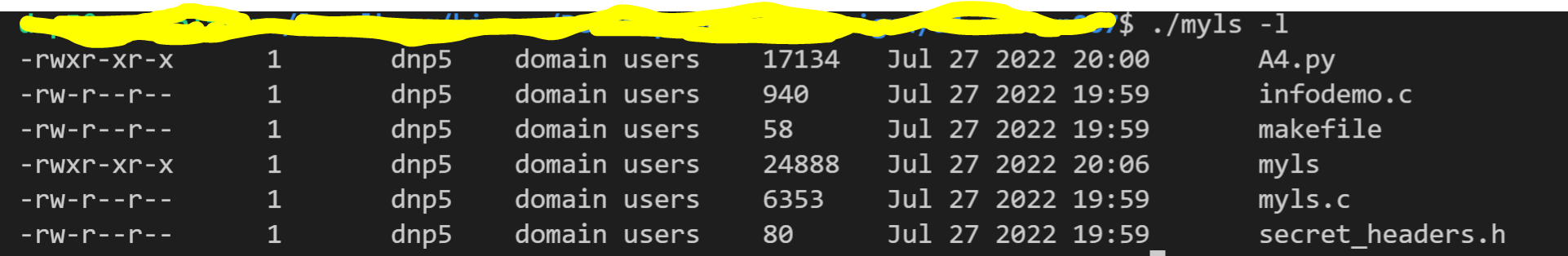






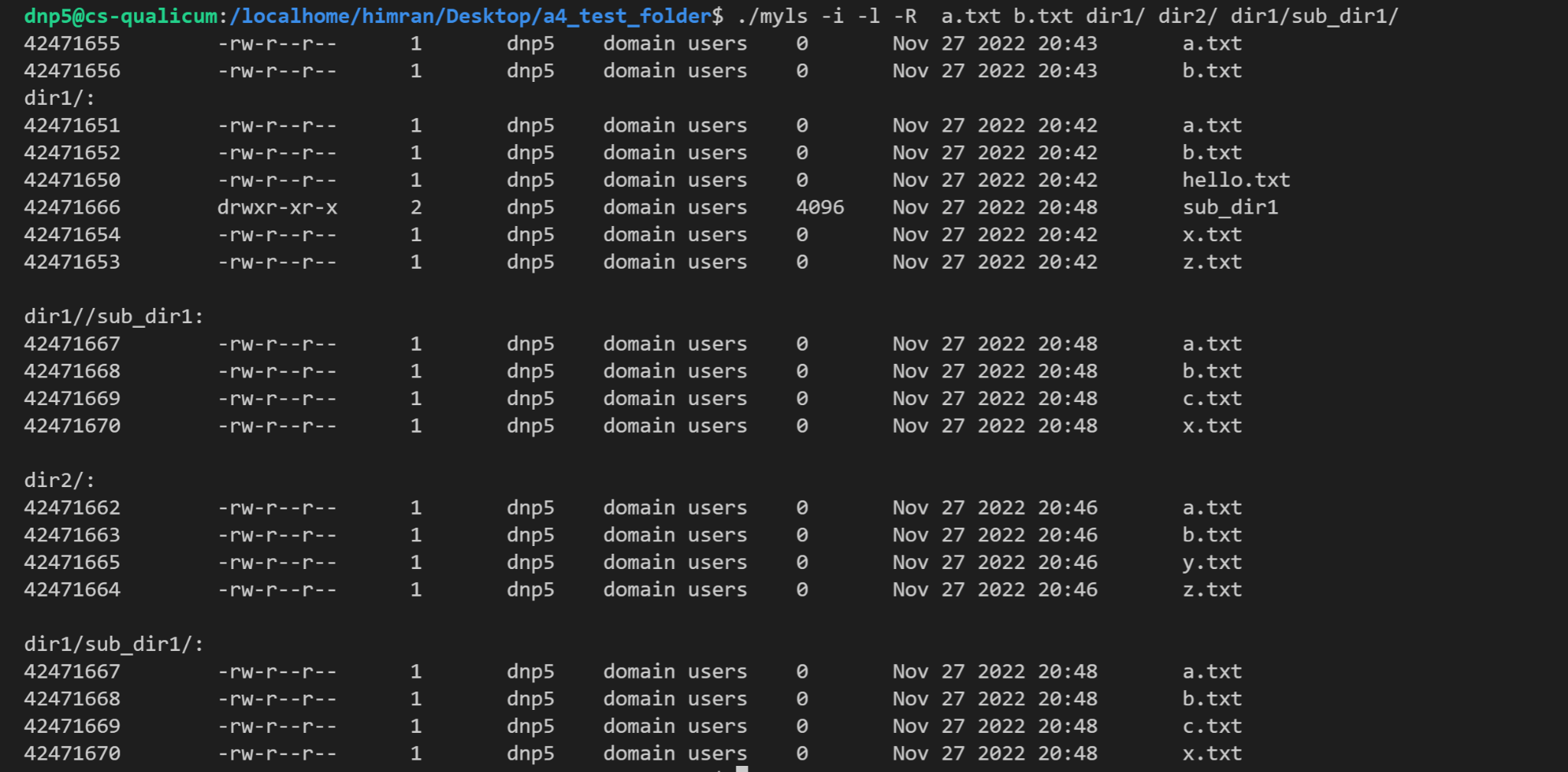


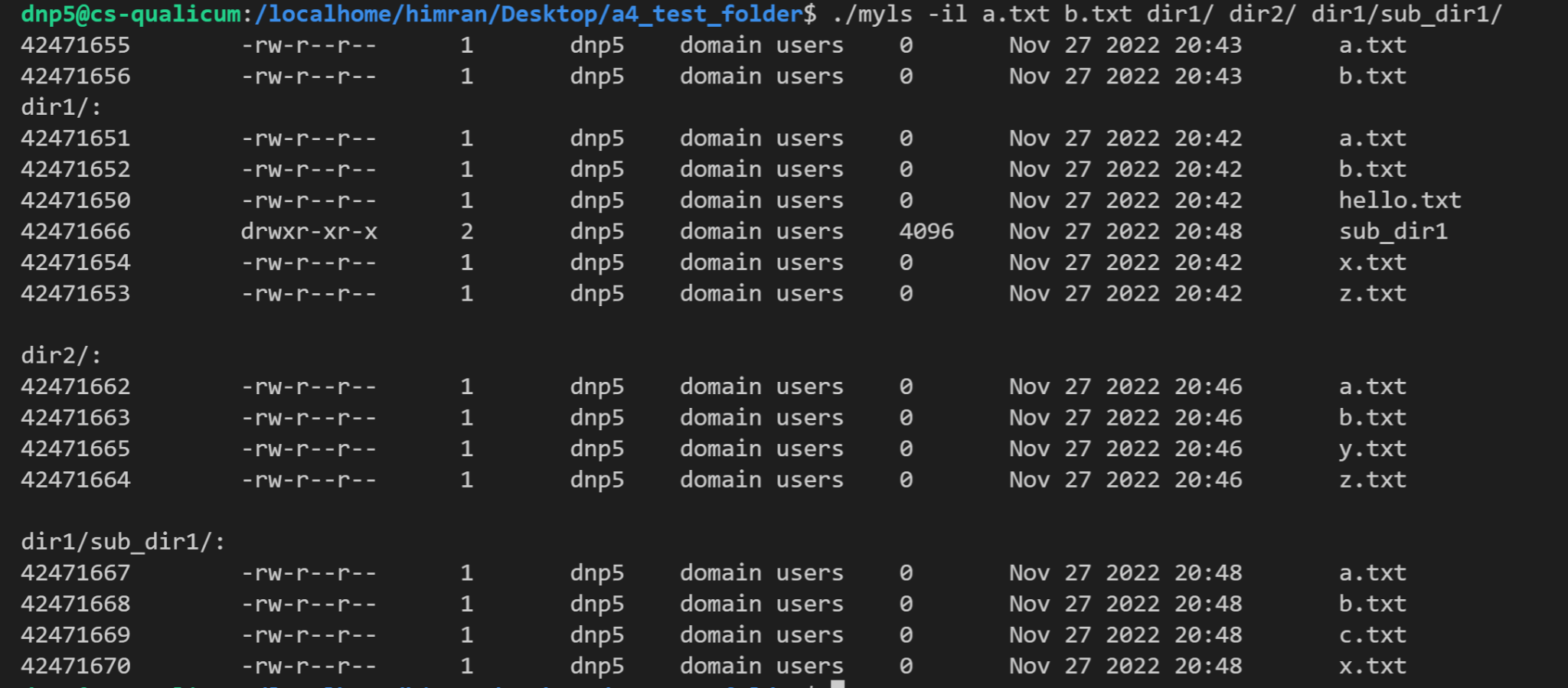


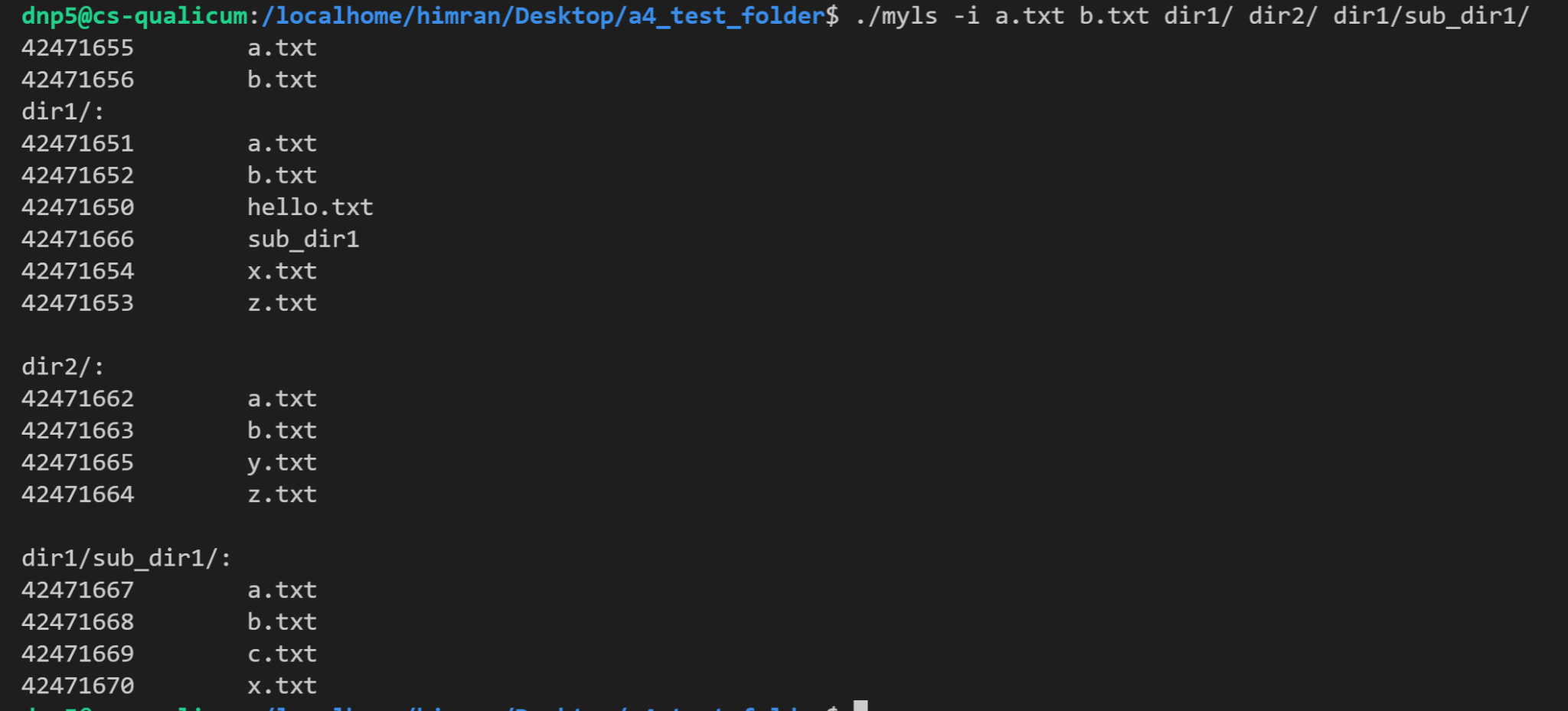


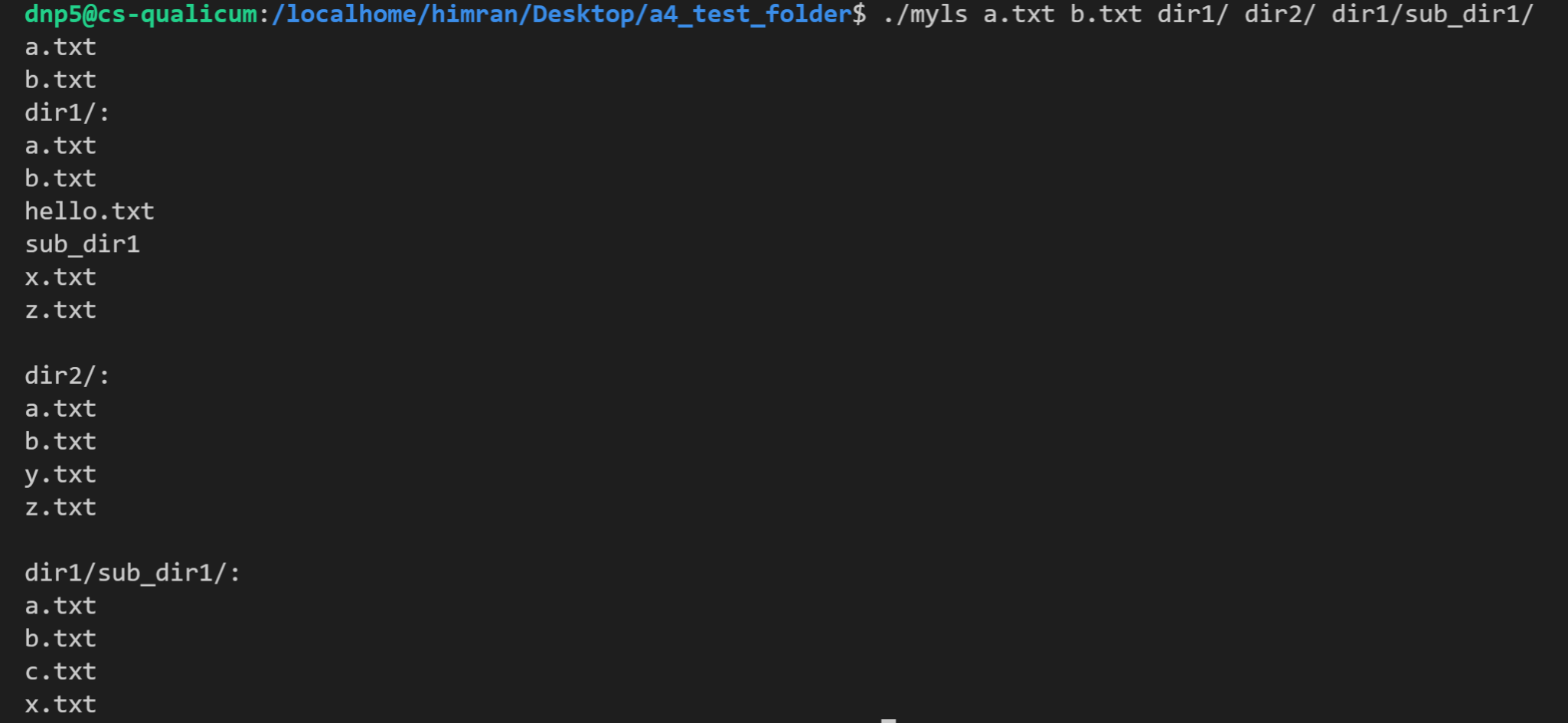
For symbolic link files use the format in the below screenshot

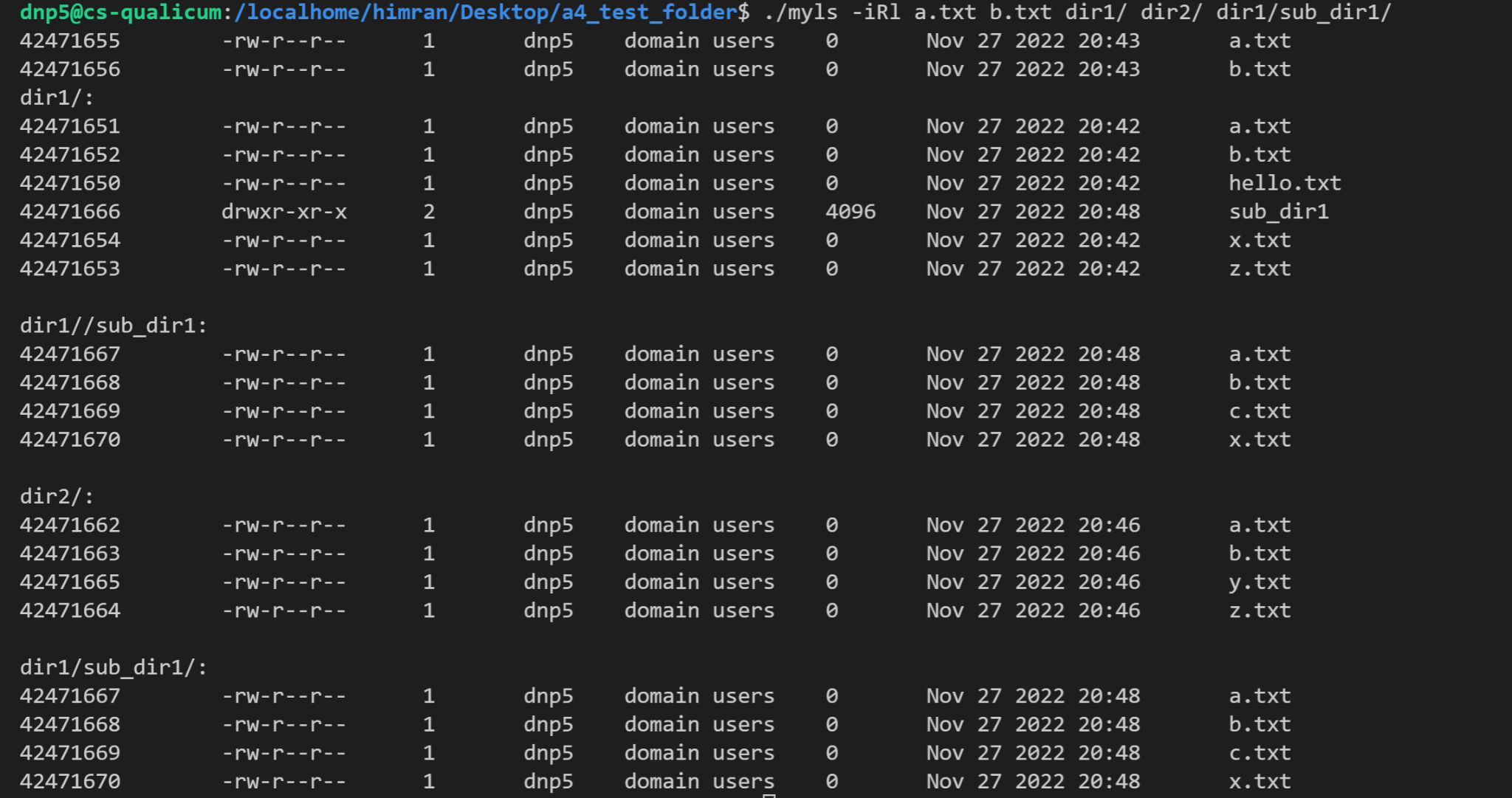
For wildcard example:

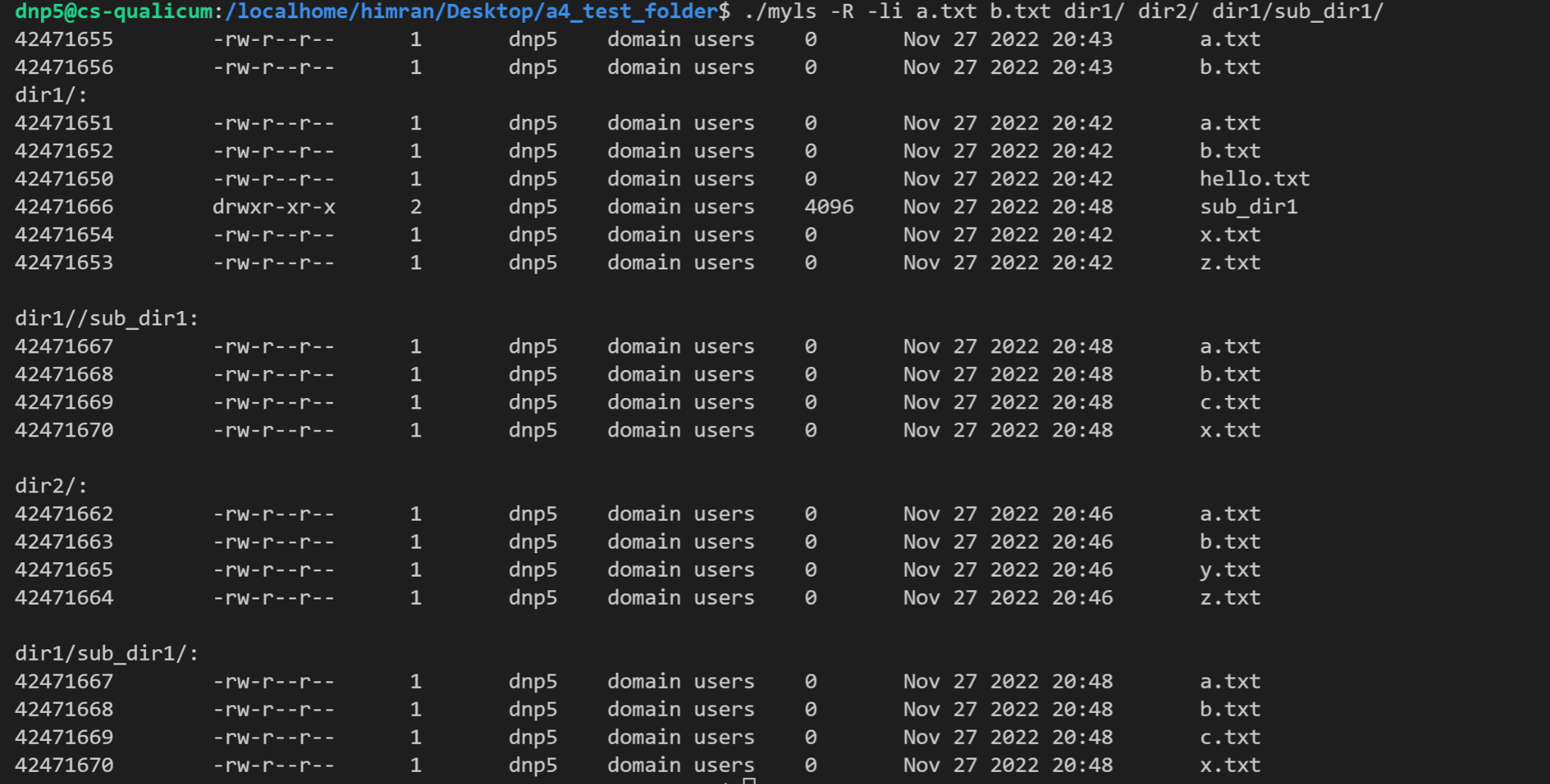
Screenshots which includes combinations of **files, directory and sub-directory**  


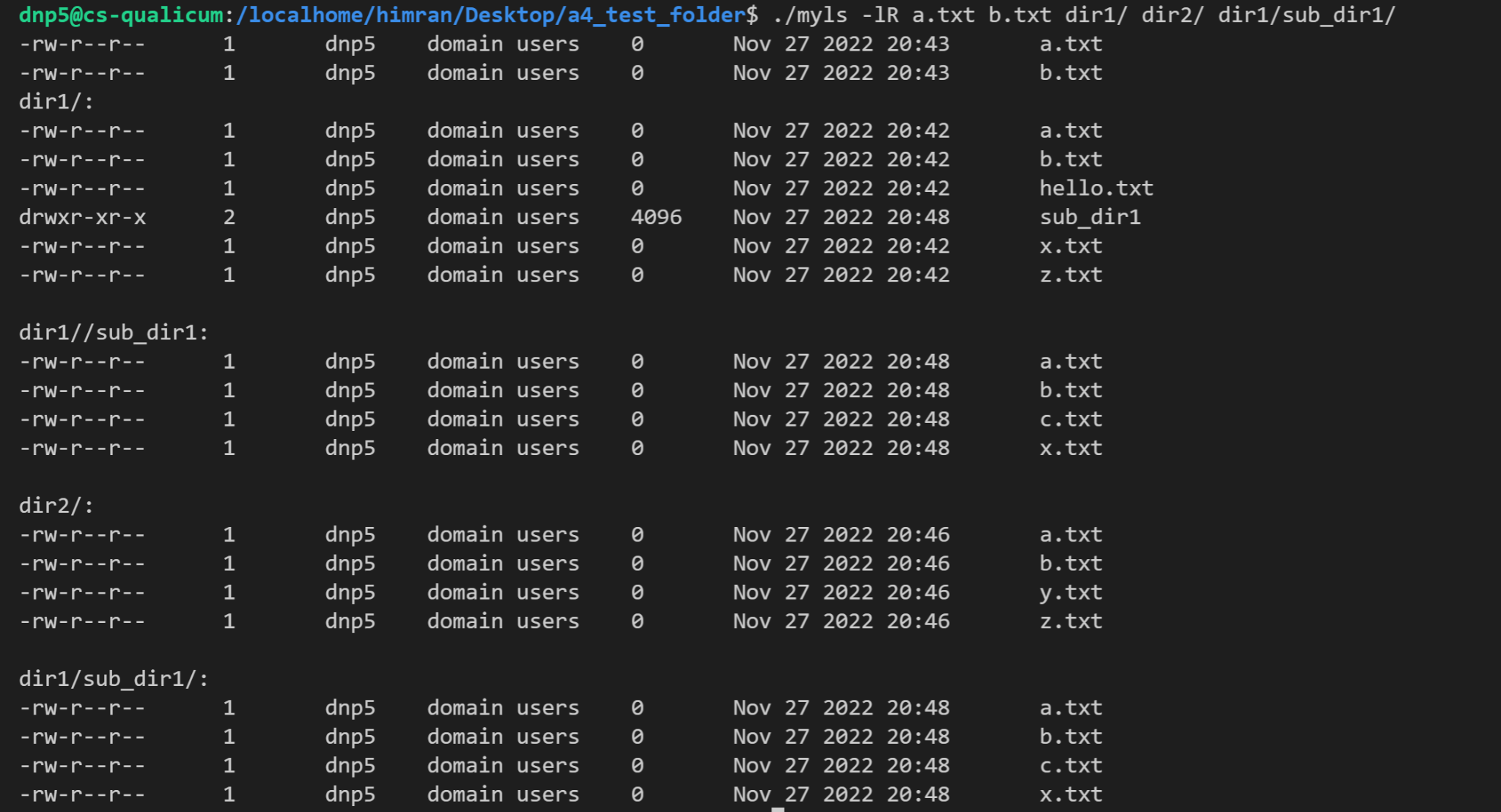


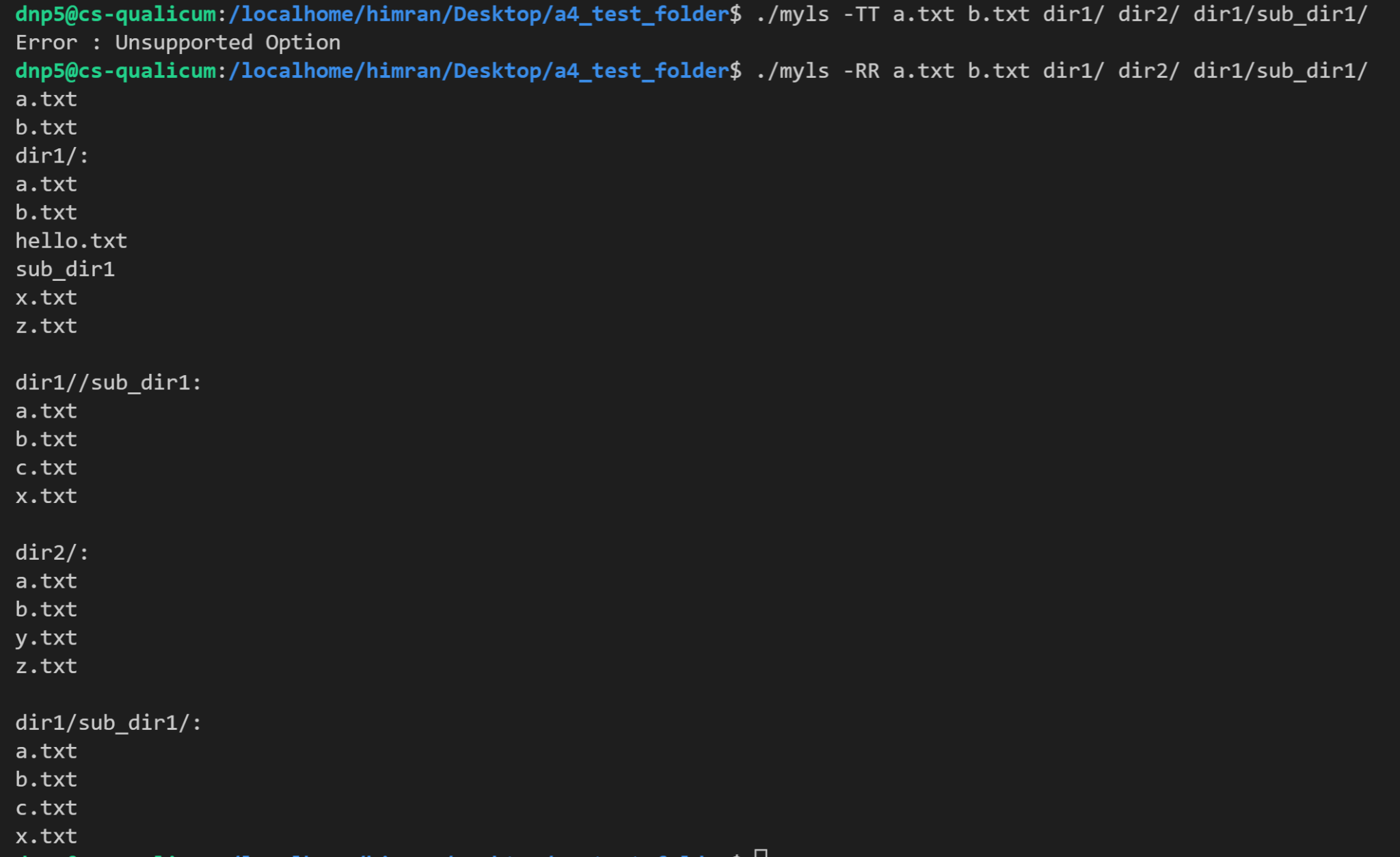












**\*\*Note: 0 after “domain users” is the size of file. Since most if the sample file is empty, file size is 0.**

FAQ

1. Can i simply run execve with a path to the program ls with the input arguments?

No

1. Can I use dirent.h for assignment 4?

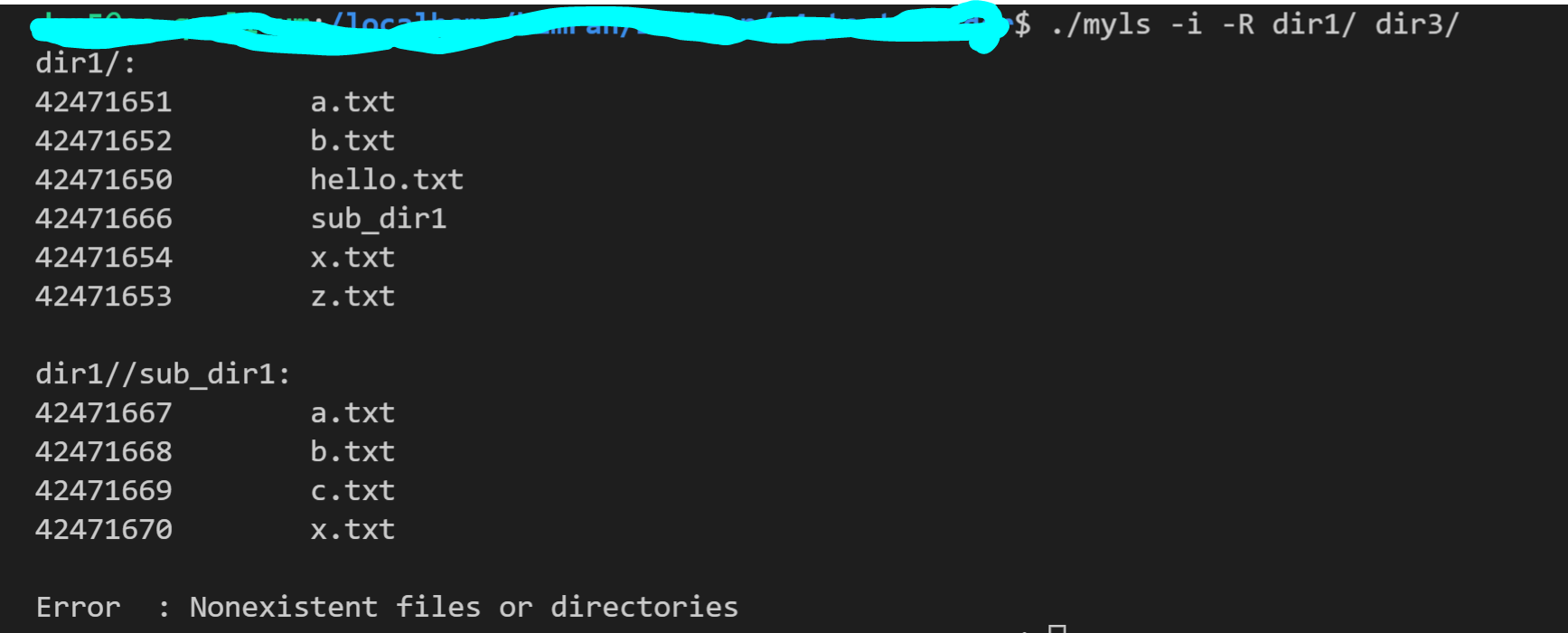
Yes

1. Can we use alphasort for -R?

Yes

1. Are the following cmd valid inputs for myls?
   1. ./myls -i -i -i -i [Valid]
   2. ./myls -i -l [Valid]
   3. ./myls -i -R [Valid]
   4. ./myls -i existing\_directory non\_existing\_directory [Not valid]

**Answer: All the above cmds are valid except last cmd. If directory doesn’t exist, you may display below error message. In the below sample dir3 doesn’t exist.**

****