# Operating Systems – spring 2022
# Assignment 9

Instructor: Hans P. Reiser

**Submission Deadline: Monday, March 21, 2022 – 23:59**

A new assignment will be published every week, right after (or a bit before) the last one was due. It must be completed before its submission deadline. (Hard deadlines, no extension.)

**T-Questions** are theory homework assignments. **The answers to the assignments must be uploaded to Canvas (in the quiz).**

**P-Questions** are programming assignments. Download the provided template from Canvas. Do not fiddle with the compiler flags. Submission instructions can also be found in first assignment.

The topic of this assignment is the POSIX API for file and directory access.

## T-Question 9.1: Files

a. *Sparse* files have unallocated holes, which correspond to zeros (i.e., the holes do not occupy space on disk; when read, they read as bytes with value zero). When a write is performed to an offset in a hole, a new block is allocated that can hold the written data. How can you determine if a file is a sparse file (i.e., it has holes), using the `stat` system call? **2 T-pt**

   *Hint:* You are expected to research the answer to this question on your own. Take a look at the man page of `stat` and the fields in `struct stat`.

b. What are ACLs? **1 T-pt**

c. Assume you have to following permissions on a set of files: **3 T-pt**

```
$ ls -lad folder folder/*
drwx--x--x. 2 hansr users 43 Mar 12 17:26 folder
-rw-r--r--. 1 hansr users  6 Mar 12 17:24 folder/example.txt
-rw-rw-rw-. 1 hansr users  6 Mar 12 17:26 folder/example2.txt
```

   Can the user `anna`, in group `users`, perform the following actions (assuming that `anna` can access everything between the root of the file system and `folder`, and that no additional ACLs are used)?

| true | false | |
|---|---|---|
| ☐ | ☐ | List the contents of `folder` |
| ☐ | ☐ | Append data at the end of the file `example2.txt` |
| ☐ | ☐ | Delete the file `example2.txt` |
| ☐ | ☐ | Execute `folder` as program in a child process |
| ☐ | ☐ | Read the content of the file `example.txt` |
| ☐ | ☐ | Read the meta data (file size, access time stamps, etc.) of `example.txt` |

d. Assume you execute the sequence of commands below on a shell. What will be the link count of `test.txt`, `b.txt`, `c.txt`, and `link.txt` (answer with 0 if not applicable/file does not exist)? **2 T-pt**

```
echo "Hello" > test.txt
ln test.txt b.txt
ln -s test.txt c.txt
ln c.txt link.txt
rm c.txt
```

e. After execution of the commands of the previous question, you execute the two commands below. What will be the content of `link.txt` and `b.txt`? **2 T-pt**

```
rm test.txt
echo "Bye" > test.txt
```

## P-Question 9.1: Directory Listing

Download the template **p1** for this assignment from Canvas. You may only modify and upload the file `ls.c`. In this question you will write a program that prints a directory listing just like the 'ls' and 'dir' commands in Linux and Windows respectively.

a. Write a function that prints a listing of all files in the directory specified by `path`. Your function should fulfill the following requirements: **6 P-pt**

- Performs an enumeration of all files in the given directory using the `opendir()`, `readdir()`, and `closedir()` system calls.[1]
- Skips all hidden entries (i.e., file names that start with '.')
- For each file, retrieves the size in bytes and the actual size on disk in bytes using `stat()`.
- Prints the gathered information via the template's `_printLine()` function. You shall print the file name without the path (e.g., 'filename.txt' instead of '/home-/x/filename.txt').
- Returns 0 on success, -1 otherwise (in case of any error).

```
int list(const char* path);
```

## P-Question 9.2: File Copy

Download the template **p2** for this assignment from Canvas. You may only modify and upload the file `copy.c`. In this question you will write a program that copies (part of) a file.

The template contains a parser for command line arguments that accepts command lines in the form '-s <n> -c <n> <source> <destination>'. The parser puts all arguments into a `CopyArgs` structure.

The `-s` (skip) parameter is optional (default value 0) and specifies the number of bytes that you should skip at the beginning of the source file.

The `-c` (count) parameter is also optional and specifies the number of bytes to copy. (If missing, you should copy all data until the end of the source file; the command line parser sets the corresponding struct field to -1).

Example: `./copy -s 3 -c 2 hello.txt out.txt` with a file `hello.txt` containing the string '`Hello world!`' should result in a file `out.txt` containing '`lo`'.

a. Write a function that performs the actual file copy. Your function should fulfill the following requirements: **4 P-pt**

- Performs the copy using the `open()`, `lseek`, `read()`, `write()`, and `close()` system calls.
- Fails if the destination file already exists.
- Uses `buffer` as intermediate buffer (you have to handle files larger than the buffer correctly).
- Returns 0 on success, -1 otherwise.

```
int doCopy(CopyArgs *args);
```

**Total:**
**10 T-pt**
**10 P-pt**

[1]On Linux, in fact you use the POSIX library function `readdir()`, which internally uses the `getdents()` system call.