Advanced Programming in the UNIX Environment

Week 03, Segment 5: umask(2)

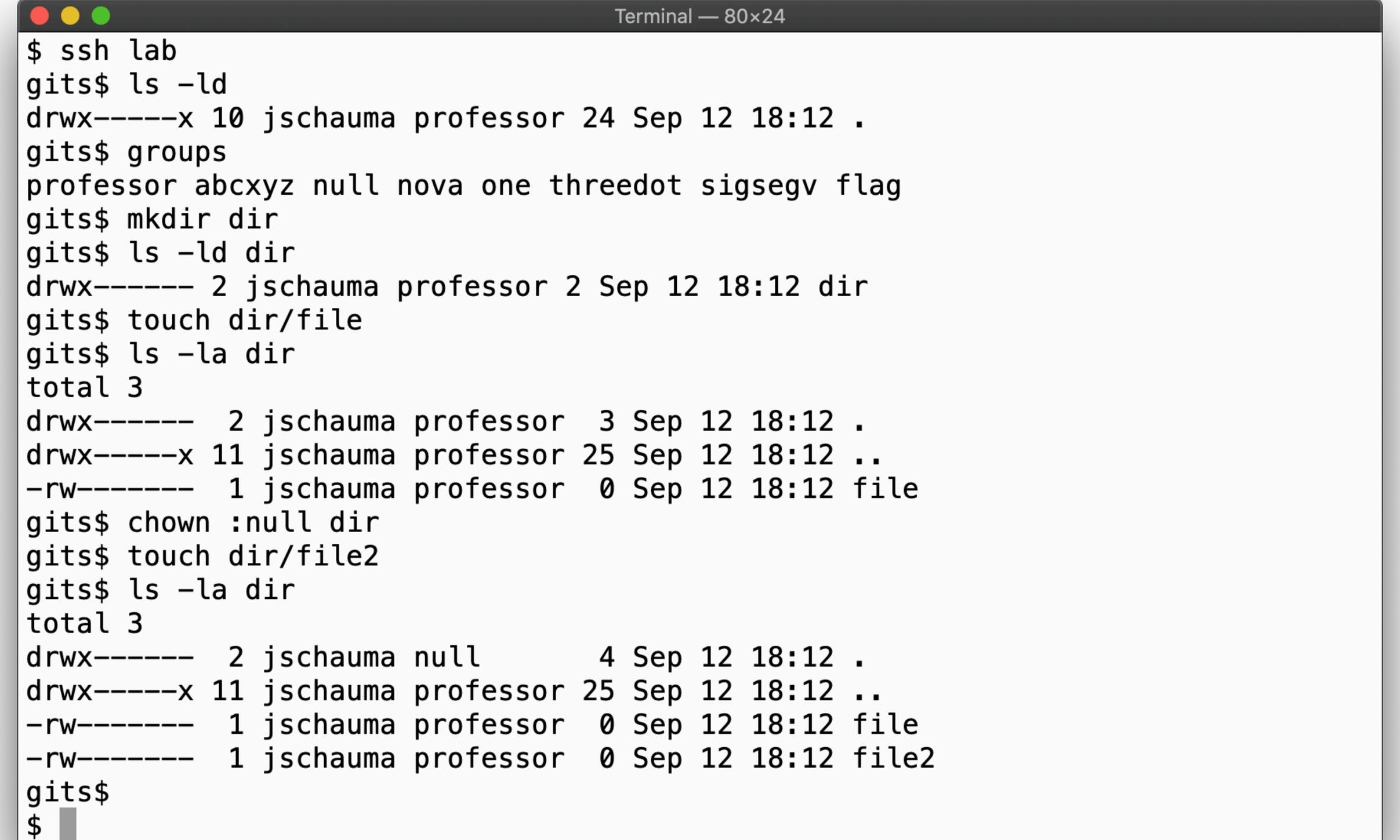
Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

Ownership of new files

When creating a new file, it will inherit:

- `st_uid` == effective UID
- `st_gid` == ... either:
  - effective GID of the process
  - GID of directory in which it is created


Terminal — 80×24
$ ssh lab
gits$ ls -ld
drwx-----x 10 jschauma professor 24 Sep 12 18:12 .
gits$ groups
professor abcxyz null nova one threedot sigsegv flag
gits$ mkdir dir
gits$ ls -ld dir
drwx------ 2 jschauma professor 2 Sep 12 18:12 dir
gits$ touch dir/file
gits$ ls -la dir
total 3
drwx------ 2 jschauma professor 3 Sep 12 18:12 .
drwx-----x 11 jschauma professor 25 Sep 12 18:12 ..
-rw------- 1 jschauma professor 0 Sep 12 18:12 file
gits$ chown :null dir
gits$ touch dir/file2
gits$ ls -la dir
total 3
drwx------ 2 jschauma null   4 Sep 12 18:12 .
drwx-----x 11 jschauma professor 25 Sep 12 18:12 ..
-rw------- 1 jschauma professor 0 Sep 12 18:12 file
-rw------- 1 jschauma professor 0 Sep 12 18:12 file2
gits$
$

umask(2)

```
#include <sys/stat.h>

mode_t umask(mode_t numask);
```

Returns: previous umask

umask(2) sets the *file creation mode mask*. Any bits that are *on* in the file creation mask are turned *off* in the file's mode.

This allows a user to set a default umask. If a program needs to be able to insure certain permissions on a file, it may need to turn off (or modify) the umask, which affects only the current process.

```
Terminal — 80×24
                perror("can't chmod file");
                exit(EXIT_FAILURE);
        }

        /* set absolute mode to rw-r--r-- */
        if (chmod("file1", S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH) == -1) {
                perror("can't chmod file1");
                exit(EXIT_FAILURE);
        }

apue$ cc chmod.c
apue$ touch file file1
apue$ ls -l file file1
-rw-------  1 jschauma  users  0 Sep 12 20:10 file
-rw-------  1 jschauma  users  0 Sep 12 20:10 file1
apue$ ./a.out
apue$ ls -l file file1
--w---S---  1 jschauma  users  0 Sep 12 20:10 file
-rw-r--r--  1 jschauma  users  0 Sep 12 20:10 file1
apue$ chmod g+x file
apue$ ls -l file file1
--w---s---  1 jschauma  users  0 Sep 12 20:10 file
-rw-r--r--  1 jschauma  users  0 Sep 12 20:10 file1
apue$
```

st_mode and UIDs recap

We've learned all about permissions and file ownership, effective UIDs and GIDs vs. real UIDs and GIDs.

You should now be able to implement most of `chown(8)` and `chmod(8)`, and with what we've covered in the previous segment, `stat(1)` as well.

In fact, come to think of it, you should be able to implement `ls(1)` itself.

Let's do that!

https://stevens.netmeister.org/631/f22-midterm.html