Implementing the ls command

Module 5 self study material

Operating systems 2018

1DT044 and 1DT096

karl.marklund@it.uu.se

Uppsala University

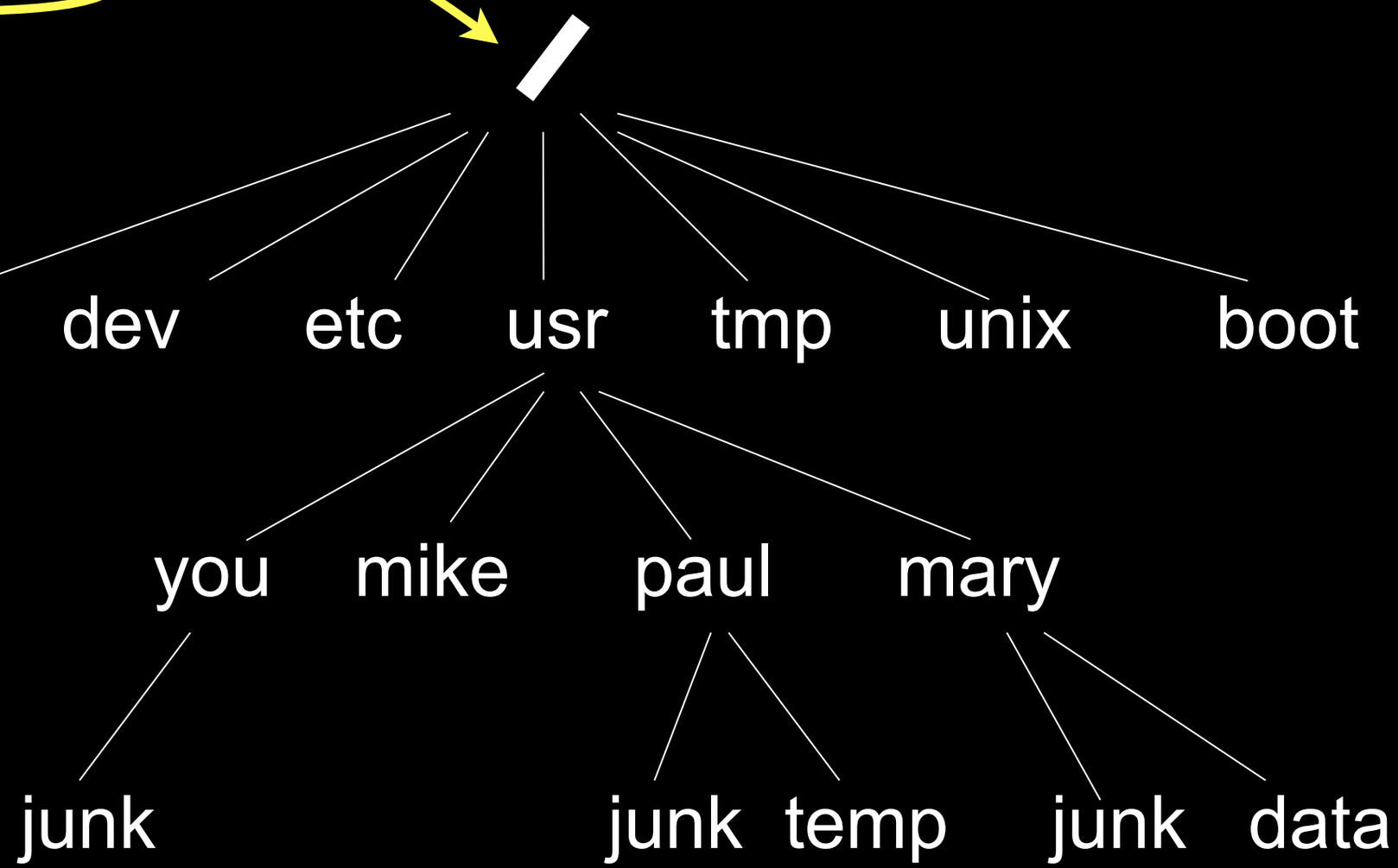
The Unix command interpreter (aka shell) does not understand the commands in any way.

Commands are implemented through system programs.

The shell creates a new process using **`fork()`** and have the new process use **`exec()`** to run the identified system program executable.

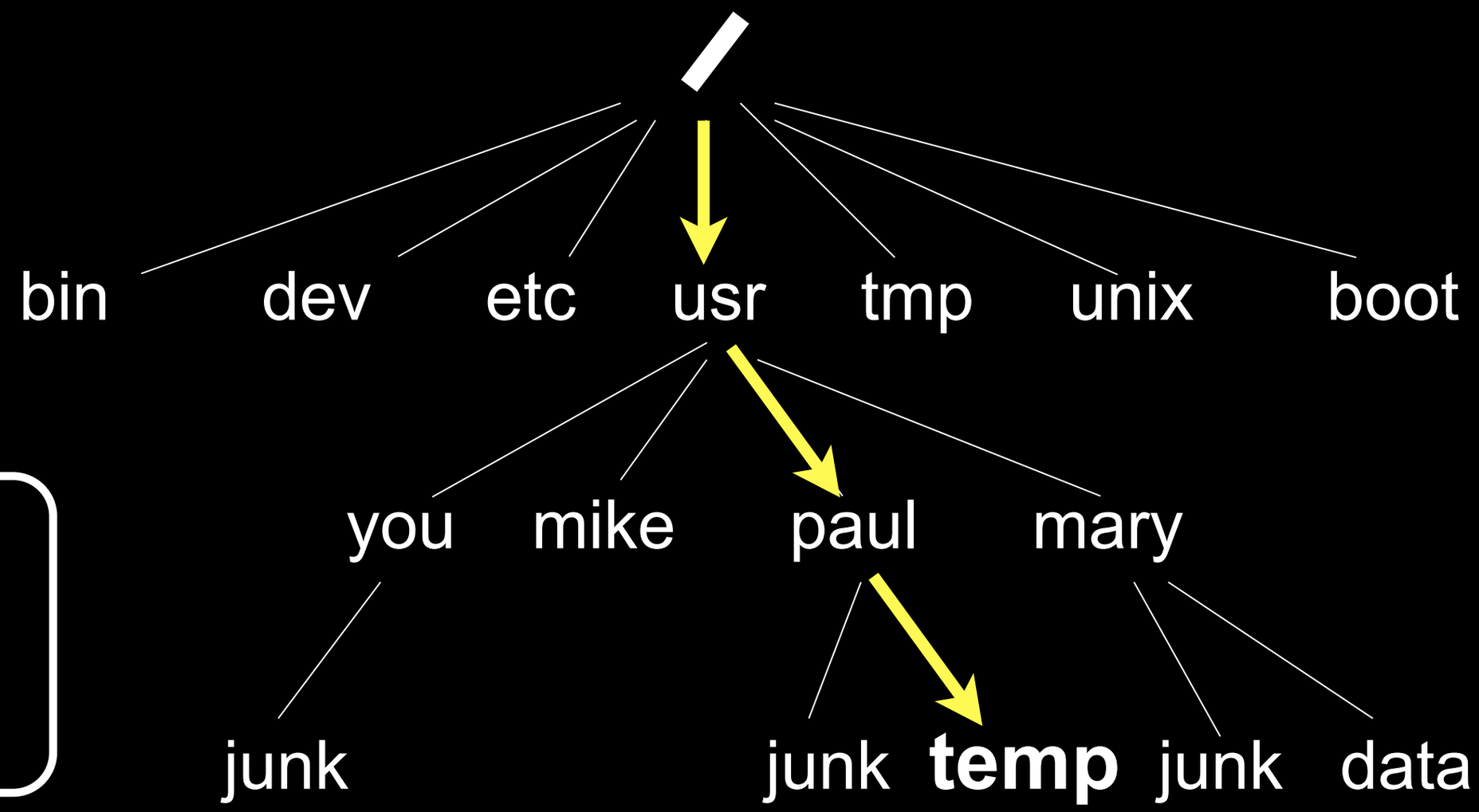
We should be able to implement our own (simplified) version of the **`ls`** command.

The Unix file system

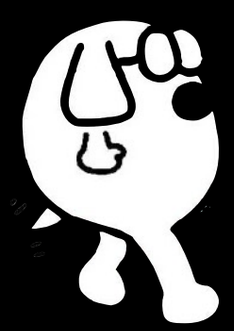


Files and directories are organized in a **tree structure** – starting at the **root**.

The Unix file system

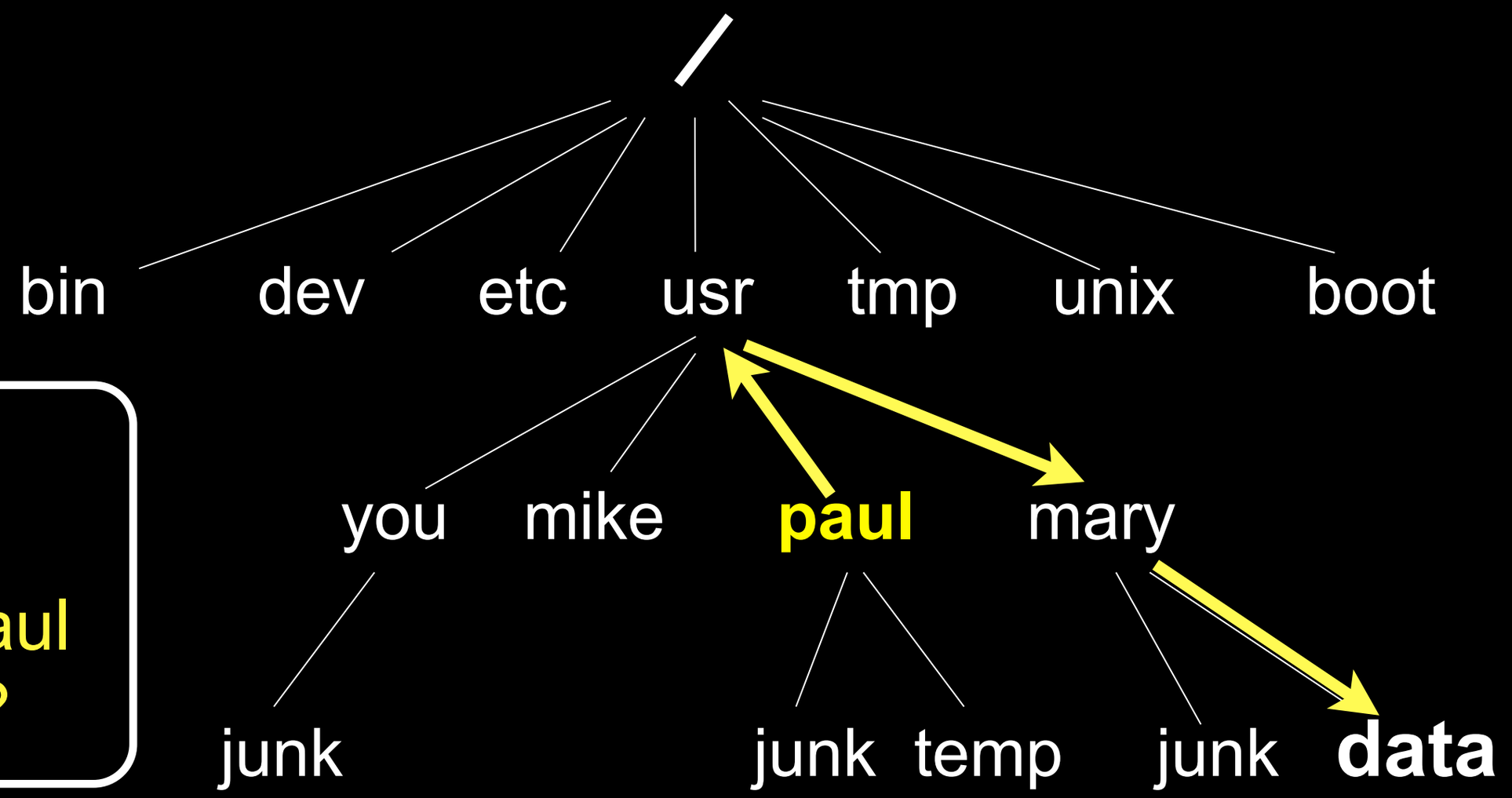


Absolute path to file temp?

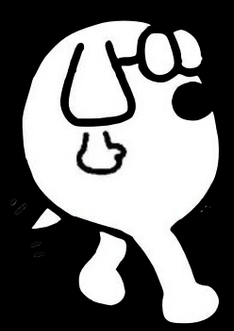


/usr/paul/temp

The Unix file system



Relative path from directory paul to file data?



```
../mary/data
```

The ls command

In Unix-like systems you can use the ls command to get information about files in a directory.

The **owner** of the file

Highest **group** that the owner belongs to

```
$> ls -l
-rw-------- 1 hans it readme.txt
-rwx-r—r-- 1 karl it script.sh
```

File mode - **permissions**

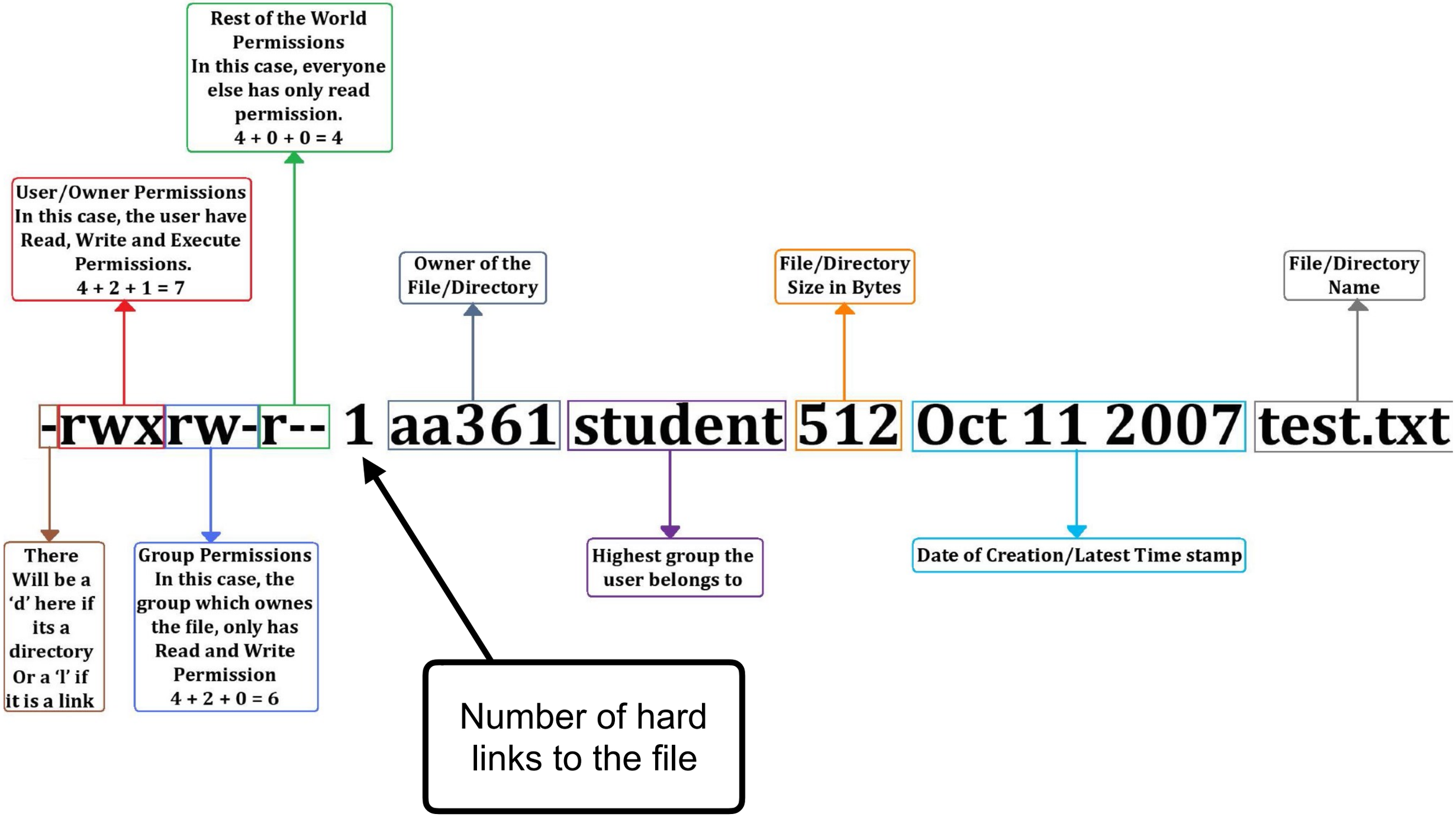
Name of File or Directory

Unix file permissions

```
$> ls -l
-rw-------  1 hans it readme.txt
-rwx-r—r--  1 karl it script.sh
```

r = read **w** = write **x** = execute

$$-\underbrace{rwx}_{\text{owner}}\underbrace{r--}_{\text{group}}\underbrace{r--}_{\text{all others}}$$


Rest of the World Permissions
In this case, everyone else has only read permission.
4 + 0 + 0 = 4
User/Owner Permissions
In this case, the user have Read, Write and Execute Permissions.
4 + 2 + 1 = 7
Owner of the File/Directory
File/Directory Size in Bytes
File/Directory Name
-rwxrw-r-- 1 aa361 student 512 Oct 11 2007 test.txt
There Will be a 'd' here if its a directory Or a 'l' if it is a link
Group Permissions
In this case, the group which ownes the file, only has Read and Write Permission
4 + 2 + 0 = 6
Highest group the user belongs to
Date of Creation/Latest Time stamp
Number of hard links to the file

In Unix-like system a file is represented by exactly one inode.

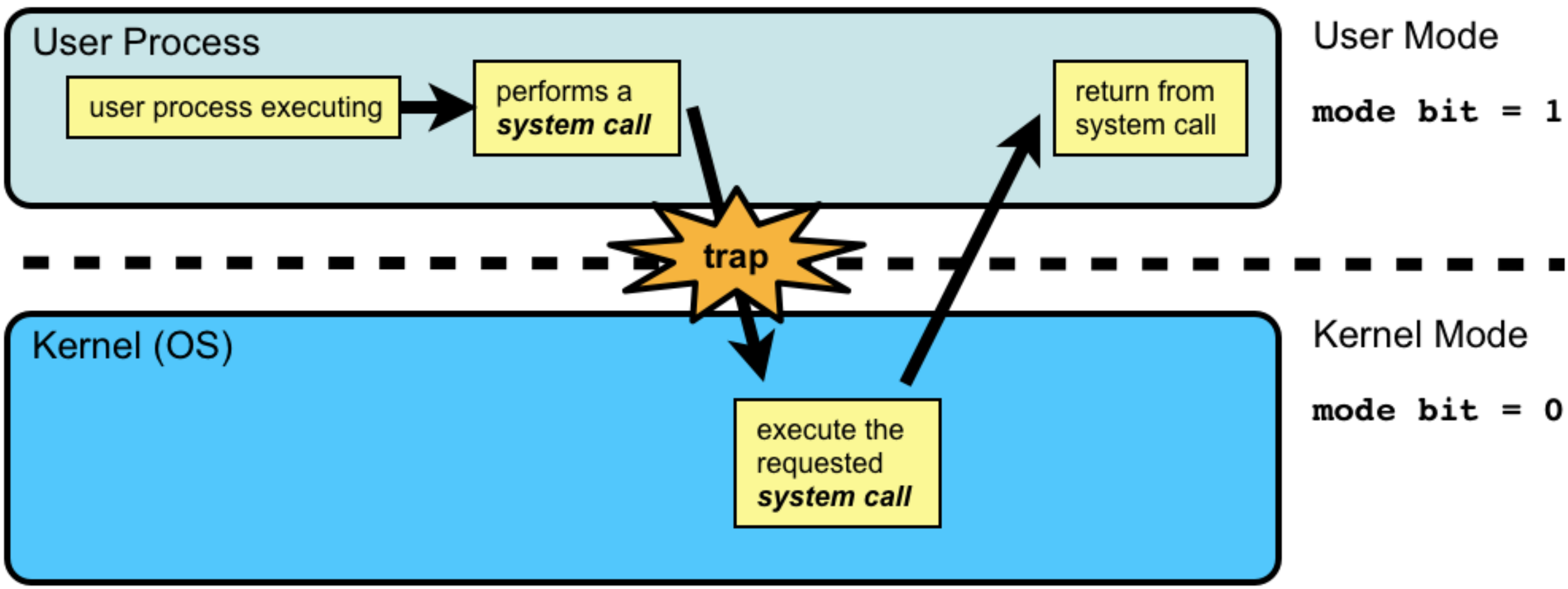
inode

Examples of information stored in an inode

File mode	**Size**
inode number	**Reference count**
Owner	**Pointer to data**
Timestamps	

- A file does not have a name.
- The file is uniquely identified by its inode number.
- The ”file name” is a property of the directory.

To obtain information about files and directories there exist a number **system calls**.



C programming with

`stat()` & `lstat()`

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

// Two system calls that can be used to
// obtain information about files or
// directories.

int  stat(const char *path, struct stat *buf);
int lstat(const char *path, struct stat *buf);

// Both stat() & lstat() returns 0 on success
// and -1 on failure.
```

Relative or absolute path to file or directory.

Must pass a **pointer** to a **stat struct** – will hold result of the system call.

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

struct stat fstat;
int status;

status = stat(”./file.txt”, &fstat);

printf(”MODE = %d \n”, (int) fstat.st_mode);

// Convert the numeric mode to the standrad
// string representation such as ”-rw-r–r–”.

printf(”MODE = %s \n”, strmode(fstat.st_mode));
```

Don’t forget to use the address-of operator & to get the pointer to `fstat`.

Now the `fstat` struct will be populated with data.

Need to implement a `strmode()` function that converts the numeric mode to string.

See the man page for the stat struct for more information

```
$ man -s2 stat
```

```
struct stat {
   dev_t     st_dev;         /* ID of device containing file */
   ino_t     st_ino;         /* inode number */
   mode_t    st_mode;        /* protection */
   nlink_t   st_nlink;       /* number of hard links */
   uid_t     st_uid;         /* user ID of owner */
   gid_t     st_gid;         /* group ID of owner */
   dev_t     st_rdev;        /* device ID (if special file) */
   off_t     st_size;        /* total size, in bytes */
   blksize_t st_blksize;     /* blocksize for filesystem I/O */
   blkcnt_t  st_blocks;      /* number of 512B blocks allocated */
   .
   .
   .
};
```

User name

How can we get the user name (string) of the file owner?

```
$> ls -l
-rw-------- 1 hans it readme.txt
-rwx-r—r-- 1 karl it script.sh
```

See the man page for the stat struct for more information

```
$ man -s2 stat
```

```
struct stat {
   dev_t     st_dev;         /* ID of device containing file */
   ino_t     st_ino;         /* inode number */
   mode_t    st_mode;        /* protection */
   nlink_t   st_nlink;       /* number of hard links */
   uid_t     st_uid;         /* user ID of owner */
   gid_t     st_gid;         /* group ID of owner */
   dev_t     st_rdev;        /* device ID (if special file) */
   off_t     st_size;        /* total size, in bytes */
   blksize_t st_blksize;     /* blocksize for filesystem I/O */
   blkcnt_t  st_blocks;      /* number of 512B blocks allocated */
   .
   .
   .
};
```

getpwuid()

Convert a numerical user id to username string.

The `getpwuid()` function shall return a pointer to a `struct passwd` with the structure as defined in `<pwd.h>` with a matching entry if found.

```
struct passwd *getpwuid(uid_t uid);
```

pwd.h

The `<pwd.h>` header shall provide a definition for `struct passwd`, which shall include at least the following members:

```
char     *pw_name   // User's login name.
uid_t     pw_uid    // Numerical user ID.
gid_t     pw_gid    // Numerical group ID.
char     *pw_dir    // Initial working directory.
char     *pw_shell  // Program to use as shell.
```

Get user name of the file owner

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>

struct stat fstat;
int status;

status = stat(”./file.txt”, &fstat);

struct passwd *pwd;

pwd = getpwuid(fstat);

printf(”USER = %s \n”, pwd->pw_name);
```

Obtaining information about links

```
$> ls -l

lrw------- 1 karl it link -> file.txt
-rw-r--r-- 1 karl it file.txt
```

- Directory entry `link` is a symbolic link to `file.txt`.
- How to find out if something is a symbolic link?
- For a symbolic link, to what does the link refer?

The difference between `stat()` & `lstat()`

```
int  stat(const char *path, struct stat *buf);

int lstat(const char *path, struct stat *buf);
```

- The `lstat()` function shall be equivalent to `stat()`, except when path refers to a symbolic link.
- In that case `lstat()` shall return information about the link, while `stat()` shall return information about the file the link references.

Obtaining information about links

```
#define MAX_LINK_LEN 20

struct stat fstat;
lstat(”link”, &fstat);

// Using lstat() we can check if a file
// is a symbolic link or not:

if (S_ISLNK(fstat.st_mode)) {
 char link_buffer[MAX_LINK_LEN];
 int len;
 len = readlink(”link”, link_buffer, MAX_LINK_LEN);
      link_buffer[len] = 0; // Null terminate
      printf(”link -> %s\n”, link_buffer);
}
```

Use `lstat()` so we can get info about symbolic links.

`S_ISLNK(m)` macro that tests whether file with mode `m` is a symbolic link or not.

`readlink()` system call used to get contents of symbolic link.