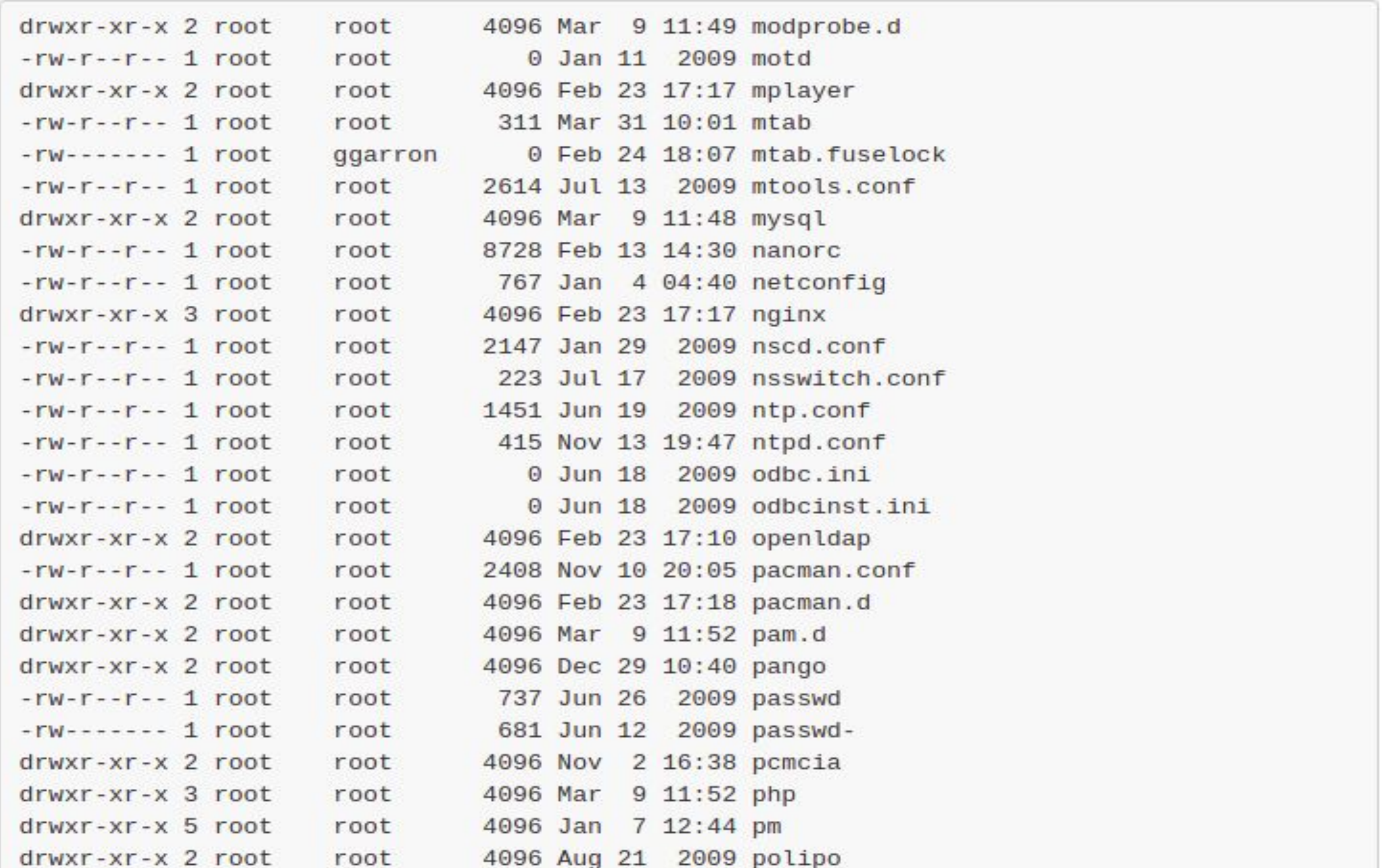
Lecture 10 & 11: Access Control

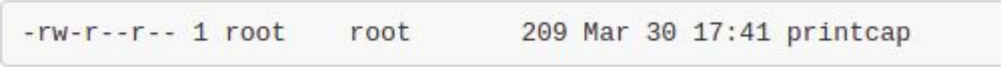
DS209 Information Security
Instructor: Dr. Souradyuti Paul

Run “ls -l”

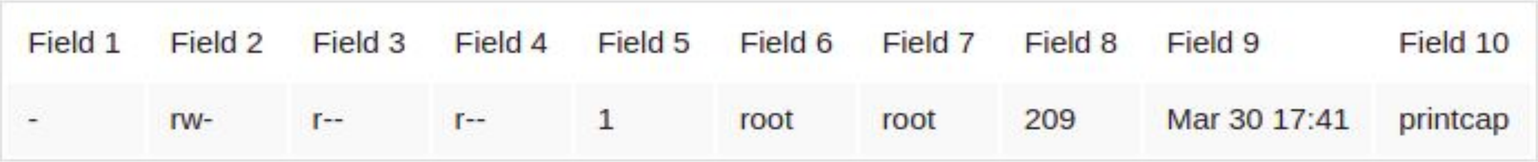


```
drwxr-xr-x 2 root    root       4096 Mar  9 11:49 modprobe.d
-rw-r--r-- 1 root    root          0 Jan 11  2009 motd
drwxr-xr-x 2 root    root       4096 Feb 23 17:17 mplayer
-rw-r--r-- 1 root    root        311 Mar 31 10:01 mtab
-rw------- 1 root    ggarron       0 Feb 24 18:07 mtab.fuselock
-rw-r--r-- 1 root    root       2614 Jul 13  2009 mtools.conf
drwxr-xr-x 2 root    root       4096 Mar  9 11:48 mysql
-rw-r--r-- 1 root    root       8728 Feb 13 14:30 nanorc
-rw-r--r-- 1 root    root        767 Jan  4 04:40 netconfig
drwxr-xr-x 3 root    root       4096 Feb 23 17:17 nginx
-rw-r--r-- 1 root    root       2147 Jan 29  2009 nscd.conf
-rw-r--r-- 1 root    root        223 Jul 17  2009 nsswitch.conf
-rw-r--r-- 1 root    root       1451 Jun 19  2009 ntp.conf
-rw-r--r-- 1 root    root        415 Nov 13 19:47 ntpd.conf
-rw-r--r-- 1 root    root          0 Jun 18  2009 odbc.ini
-rw-r--r-- 1 root    root          0 Jun 18  2009 odbcinst.ini
drwxr-xr-x 2 root    root       4096 Feb 23 17:10 openldap
-rw-r--r-- 1 root    root       2408 Nov 10 20:05 pacman.conf
drwxr-xr-x 2 root    root       4096 Feb 23 17:18 pacman.d
drwxr-xr-x 2 root    root       4096 Mar  9 11:52 pam.d
drwxr-xr-x 2 root    root       4096 Dec 29 10:40 pango
-rw-r--r-- 1 root    root        737 Jun 26  2009 passwd
-rw------- 1 root    root        681 Jun 12  2009 passwd-
drwxr-xr-x 2 root    root       4096 Nov  2 16:38 pcmcia
drwxr-xr-x 3 root    root       4096 Mar  9 11:52 php
drwxr-xr-x 5 root    root       4096 Jan  7 12:44 pm
drwxr-xr-x 2 root    root       4096 Aug 21  2009 polipo
```
]

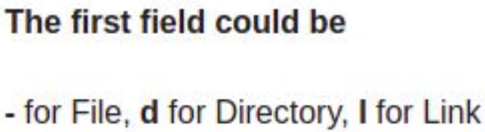
Explanation



```
-rw-r--r-- 1 root    root       209 Mar 30 17:41 printcap
```

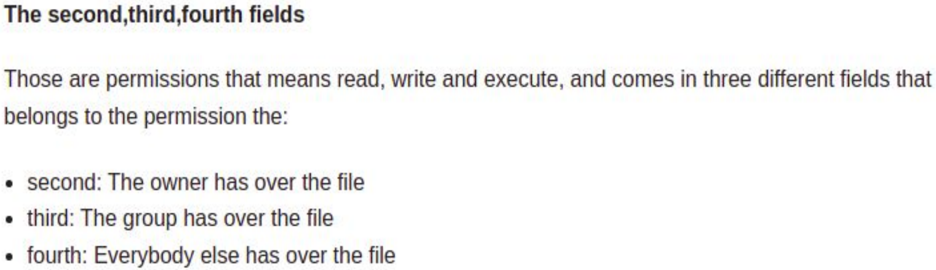


Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10
-	rw-	r--	r--	1	root	root	209	Mar 30 17:41	printcap



The first field could be

- for File, **d** for Directory, **l** for Link

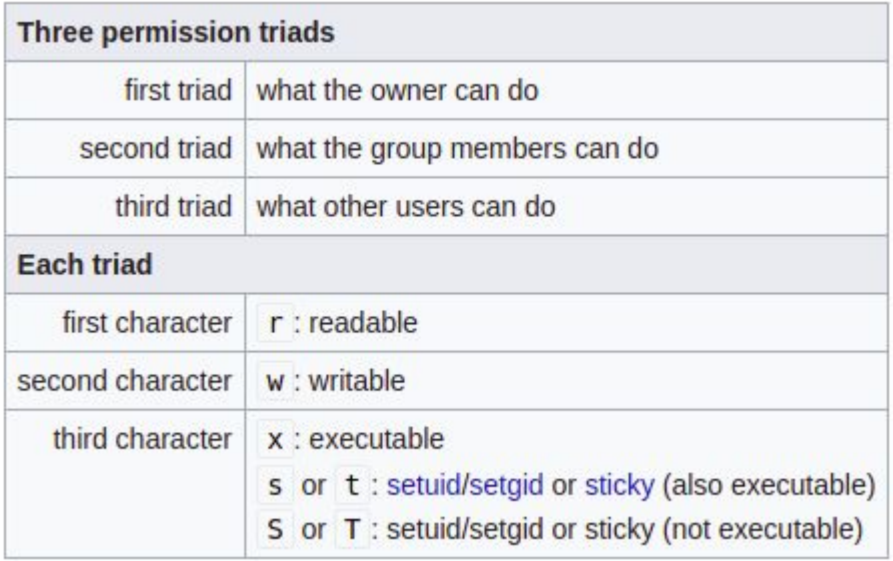


The second,third,fourth fields

Those are permissions that means read, write and execute, and comes in three different fields that belongs to the permission the:

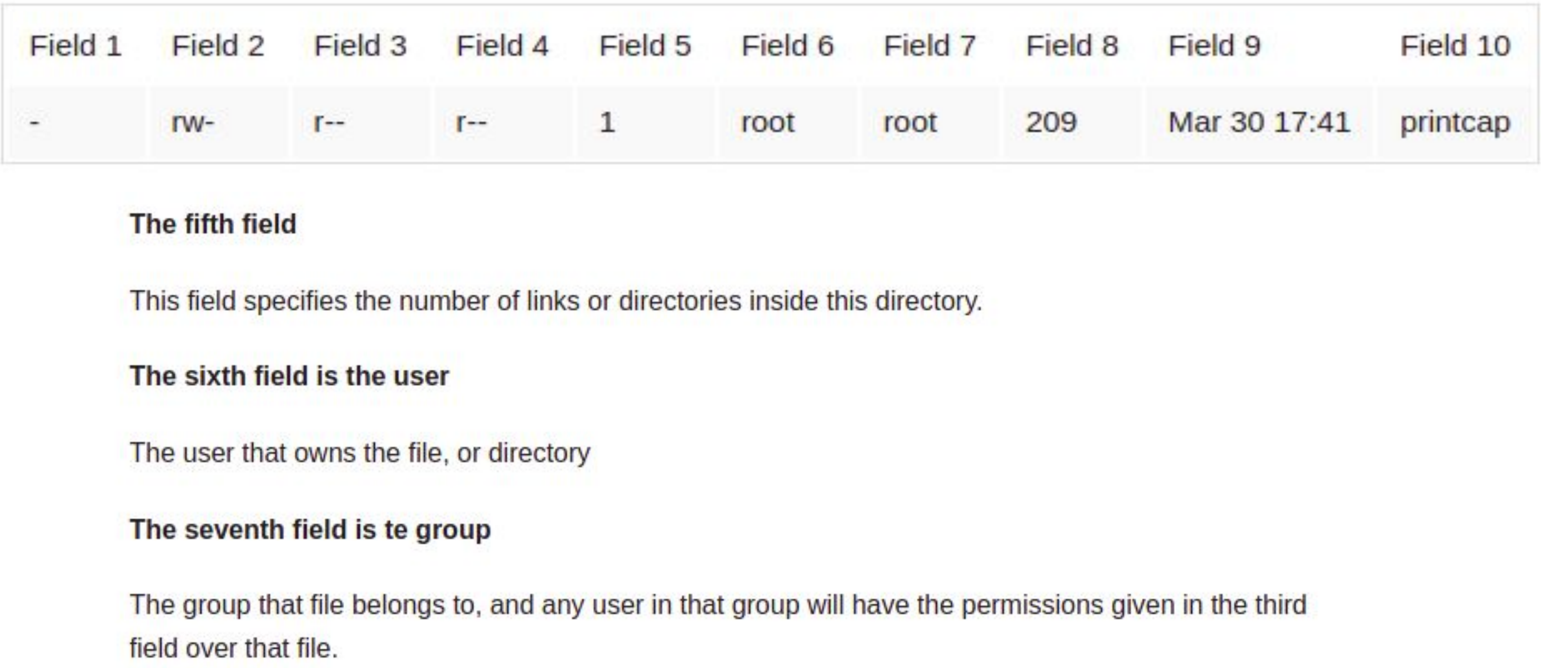
- second: The owner has over the file
- third: The group has over the file
- fourth: Everybody else has over the file

Three permission triads



Three permission triads	
first triad	what the owner can do
second triad	what the group members can do
third triad	what other users can do
Each triad	
first character	`r` : readable
second character	`w` : writable
third character	`x` : executable `s` or `t` : setuid/setgid or sticky (also executable) `S` or `T` : setuid/setgid or sticky (not executable)

5th, 6th and 7th fields



Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10
-	rw-	r--	r--	1	root	root	209	Mar 30 17:41	printcap

The fifth field

This field specifies the number of links or directories inside this directory.

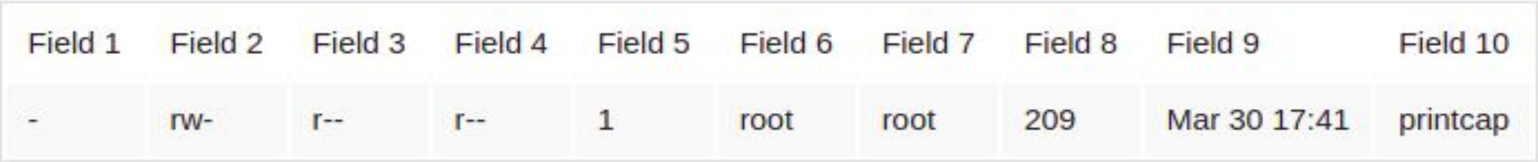
The sixth field is the user

The user that owns the file, or directory

The seventh field is te group

The group that file belongs to, and any user in that group will have the permissions given in the third field over that file.

8th, 9th and 10th fields



Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10
-	rw-	r--	r--	1	root	root	209	Mar 30 17:41	printcap

The eighth field

The size in bytes,

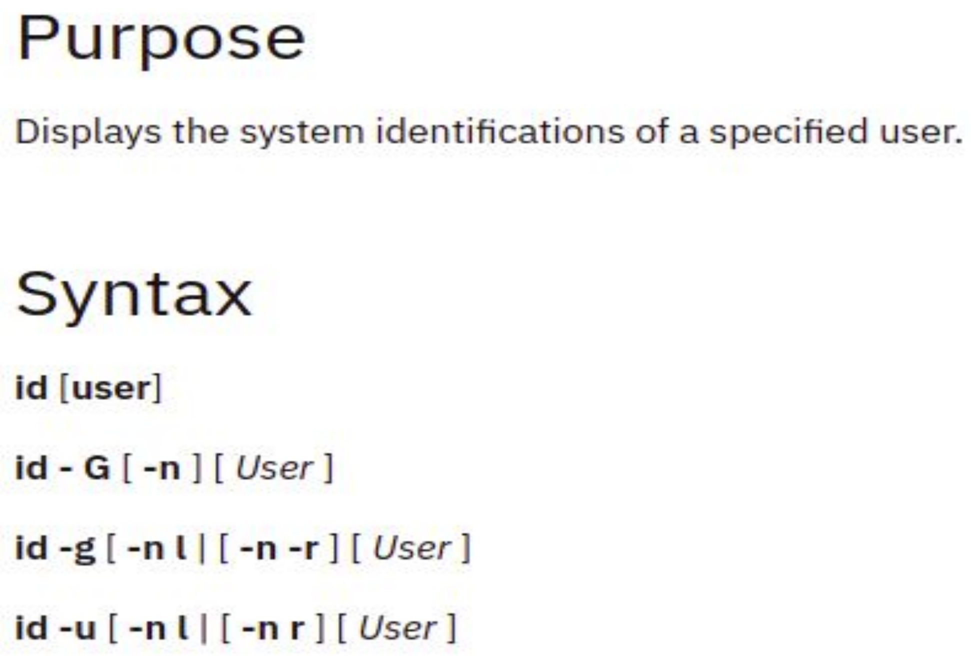
The ninth field

The date of last modification

The tenth field

The name of the file

Id command (located at: /usr/bin/id)



Purpose

Displays the system identifications of a specified user.

Syntax

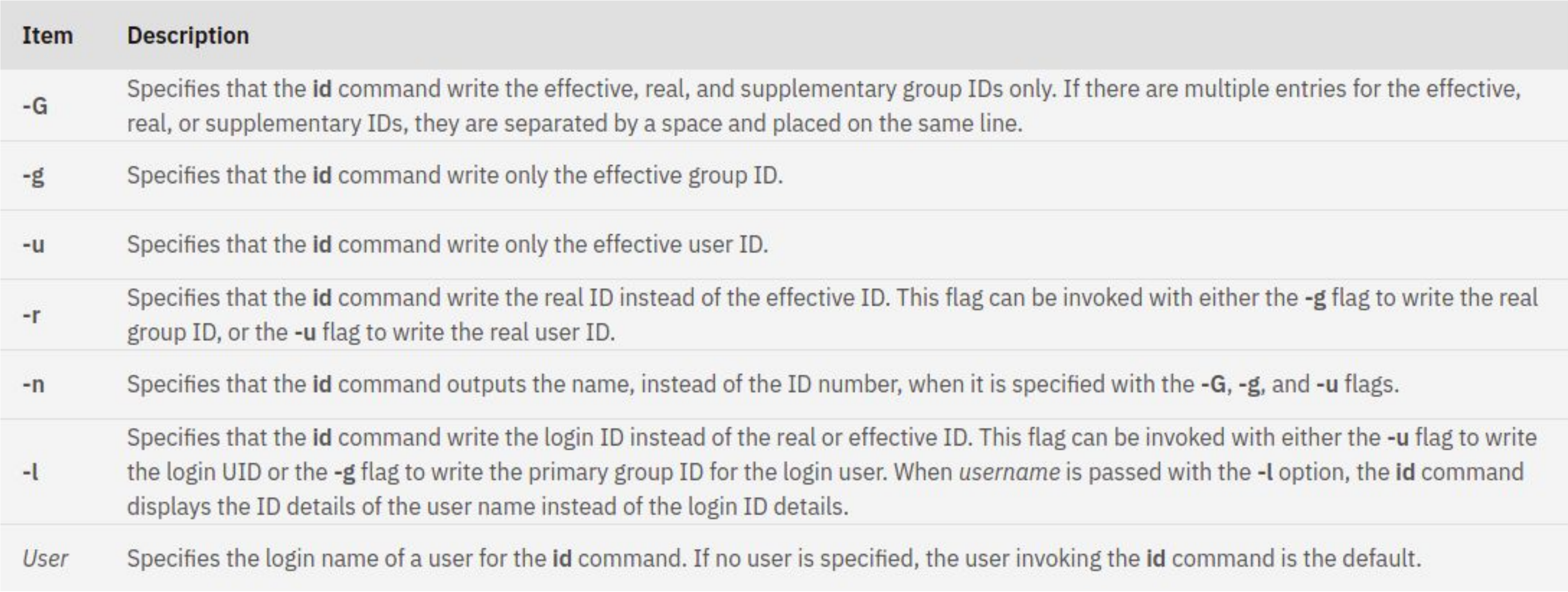
id [**user**]

id - G [**-n**] [*User*]

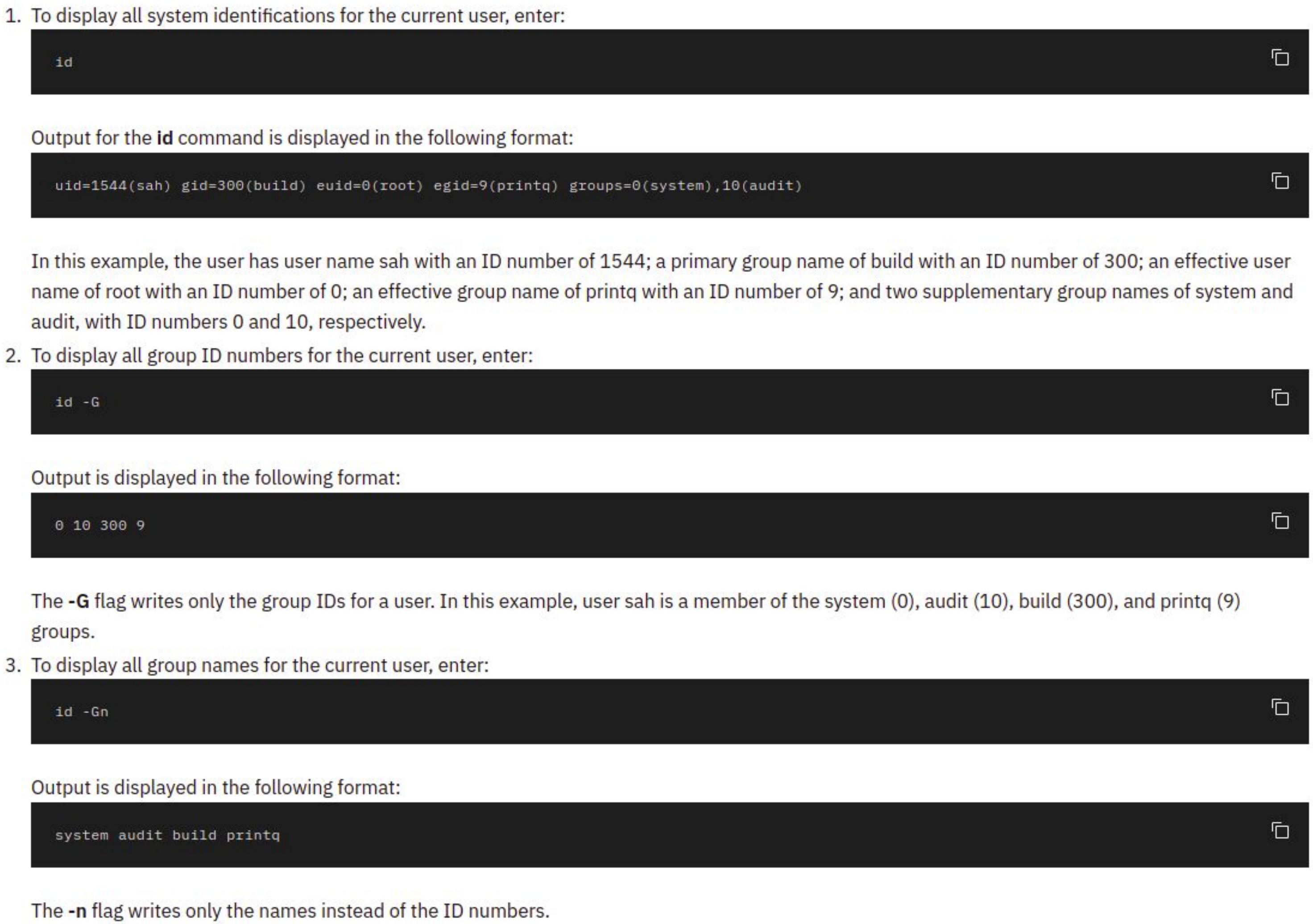
id -g [**-n l** | [**-n -r**] [*User*]

id -u [**-n l** | [**-n r**] [*User*]

Flags for id command



Item	Description
-G	Specifies that the **id** command write the effective, real, and supplementary group IDs only. If there are multiple entries for the effective, real, or supplementary IDs, they are separated by a space and placed on the same line.
-g	Specifies that the **id** command write only the effective group ID.
-u	Specifies that the **id** command write only the effective user ID.
-r	Specifies that the **id** command write the real ID instead of the effective ID. This flag can be invoked with either the **-g** flag to write the real group ID, or the **-u** flag to write the real user ID.
-n	Specifies that the **id** command outputs the name, instead of the ID number, when it is specified with the **-G**, **-g**, and **-u** flags.
-l	Specifies that the **id** command write the login ID instead of the real or effective ID. This flag can be invoked with either the **-u** flag to write the login UID or the **-g** flag to write the primary group ID for the login user. When *username* is passed with the **-l** option, the **id** command displays the ID details of the user name instead of the login ID details.
User	Specifies the login name of a user for the **id** command. If no user is specified, the user invoking the **id** command is the default.



1. To display all system identifications for the current user, enter:

   ```
   id
   ```

   Output for the **id** command is displayed in the following format:

   ```
   uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq) groups=0(system),10(audit)
   ```

   In this example, the user has user name sah with an ID number of 1544; a primary group name of build with an ID number of 300; an effective user name of root with an ID number of 0; an effective group name of printq with an ID number of 9; and two supplementary group names of system and audit, with ID numbers 0 and 10, respectively.

2. To display all group ID numbers for the current user, enter:

   ```
   id -G
   ```

   Output is displayed in the following format:

   ```
   0 10 300 9
   ```

   The **-G** flag writes only the group IDs for a user. In this example, user sah is a member of the system (0), audit (10), build (300), and printq (9) groups.

3. To display all group names for the current user, enter:

   ```
   id -Gn
   ```

   Output is displayed in the following format:

   ```
   system audit build printq
   ```

   The **-n** flag writes only the names instead of the ID numbers.

4. To display the real group name for the current user, enter:

   ```
   id -gnr
   ```

   Output is displayed in the following format:

   ```
   build
   ```

5. To display the login UID after logging in as root and running the **su** command to user **sah**, type:

   ```
   id -lu
   ```

   Output is displayed in the following format:

   ```
   0
   ```

6. To display the primary group name of the user who actually logged in, type:

   ```
   id -lgn
   ```

   Output is displayed in the following format:

   ```
   system
   ```

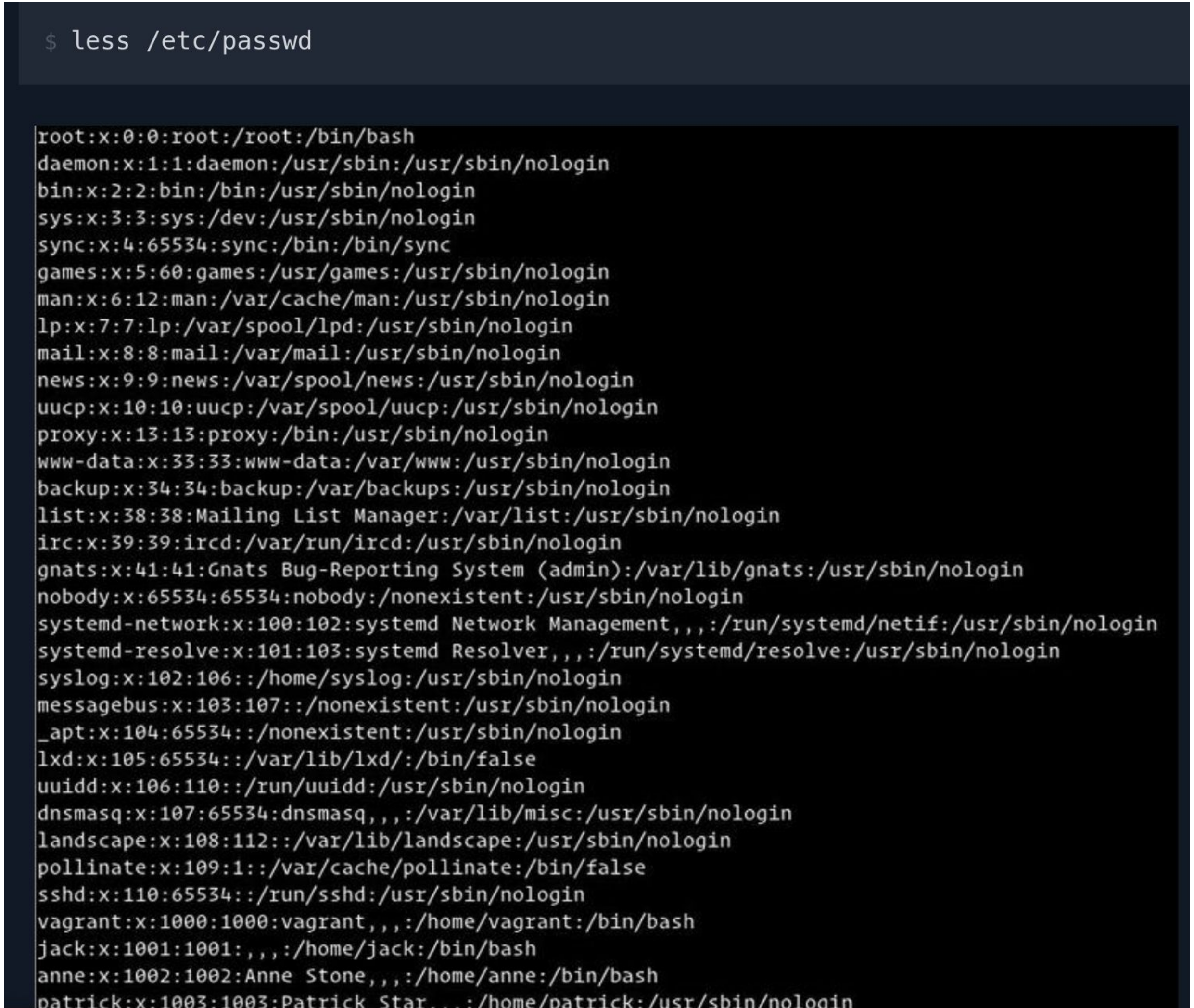
7. To display the primary group ID of the user who actually logged in, type:

   ```
   id -lg
   ```

   Output is displayed in the following format:

   ```
   0
   ```

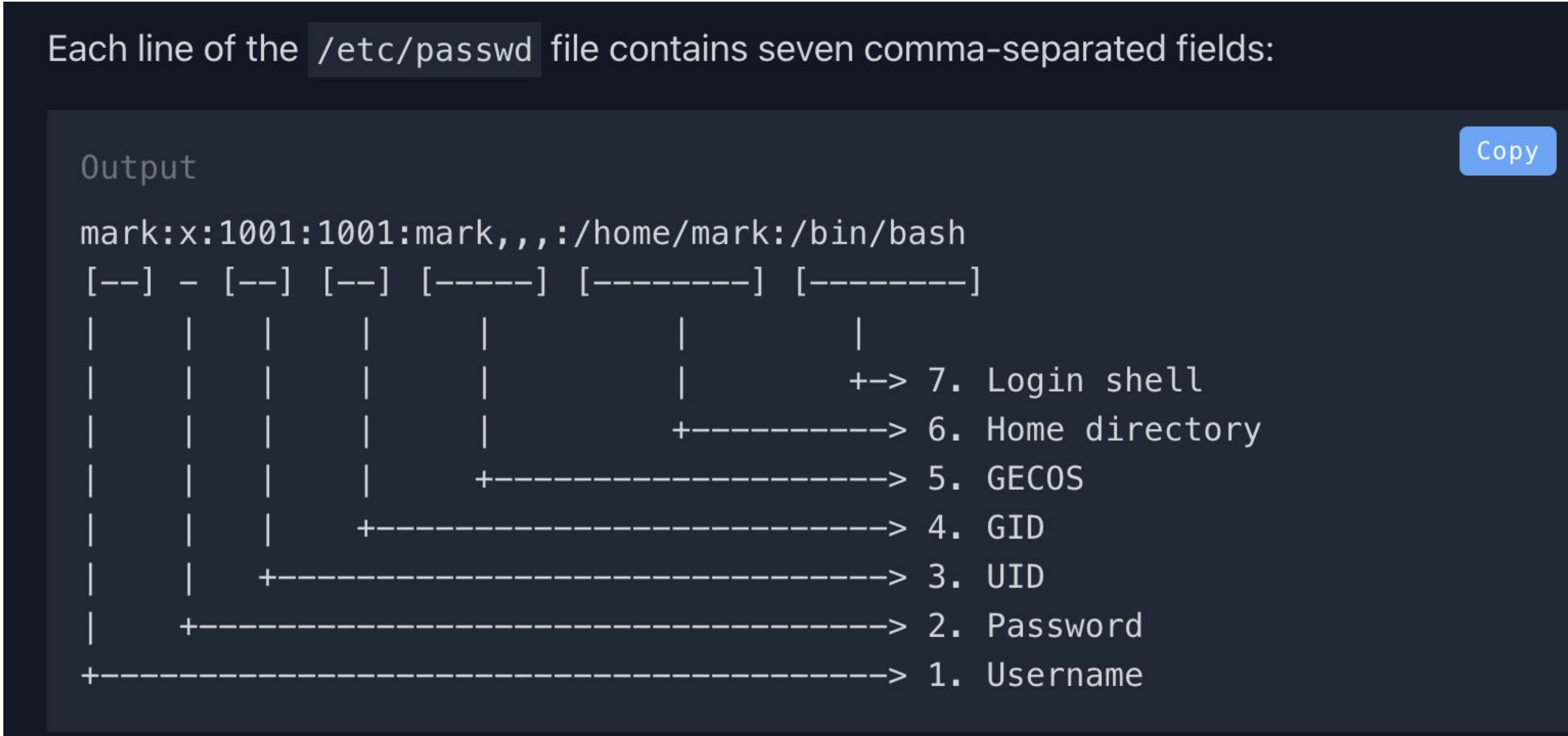
User Identification:
First understand the file at /etc/passwd
$ cat /etc/passwd or $less /etc/passwd



```
$ less /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
vagrant:x:1000:1000:vagrant,,,:/home/vagrant:/bin/bash
jack:x:1001:1001:,,,:/home/jack:/bin/bash
anne:x:1002:1002:Anne Stone,,,:/home/anne:/bin/bash
patrick:x:1003:1003:Patrick Star,,,:/home/patrick:/usr/sbin/nologin
```

Explanation (7 comma-separated fields)

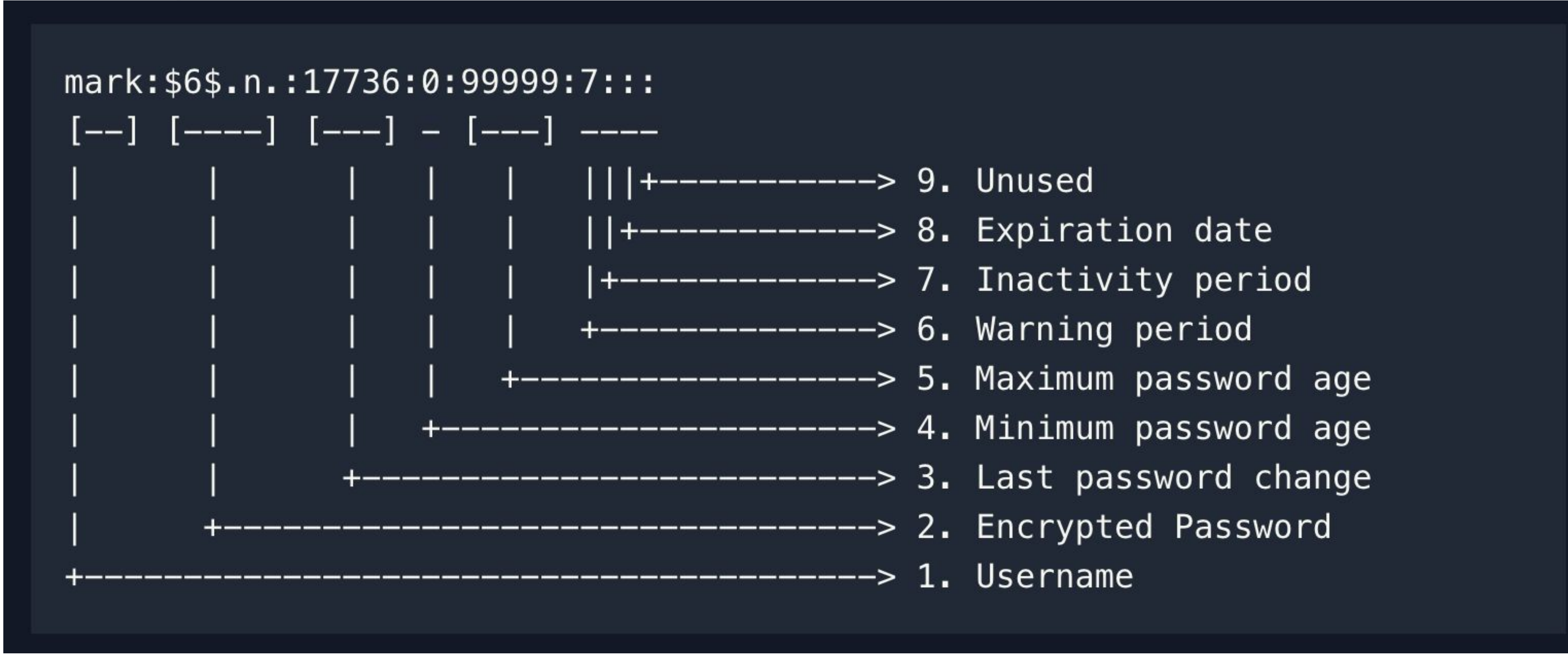


Each line of the `/etc/passwd` file contains seven comma-separated fields:

Output

```
mark:x:1001:1001:mark,,,:/home/mark:/bin/bash
[--] - [--] [--] [-----] [--------] [--------]
|    |   |    |     |         |        |
|    |   |    |     |         |        +-> 7. Login shell
|    |   |    |     |         +----------> 6. Home directory
|    |   |    |     +--------------------> 5. GECOS
|    |   |    +--------------------------> 4. GID
|    |   +-------------------------------> 3. UID
|    +-----------------------------------> 2. Password
+----------------------------------------> 1. Username
```

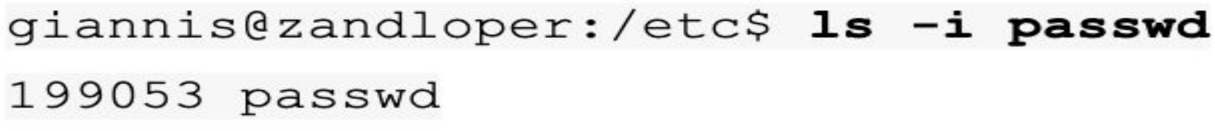
Explanation of file at /etc/shadow (9 comma-separated fields)



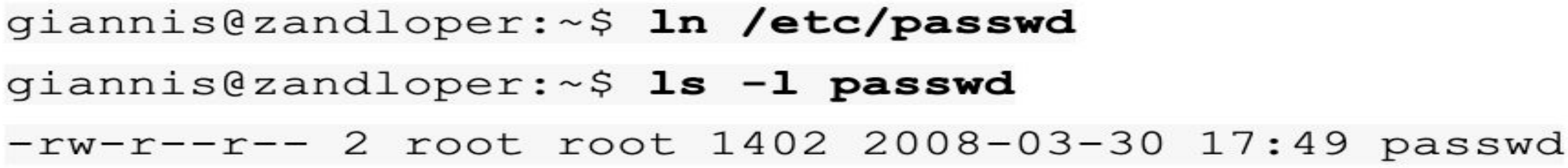
```
mark:$6$.n.:17736:0:99999:7:::
[--] [----] [---] - [---] ----
|      |      |   |   |   |||+-----------> 9. Unused
|      |      |   |   |   ||+------------> 8. Expiration date
|      |      |   |   |   |+-------------> 7. Inactivity period
|      |      |   |   |   +--------------> 6. Warning period
|      |      |   |   +------------------> 5. Maximum password age
|      |      |   +----------------------> 4. Minimum password age
|      |      +--------------------------> 3. Last password change
|      +---------------------------------> 2. Encrypted Password
+----------------------------------------> 1. Username
```

Hard link: What, How, Where?

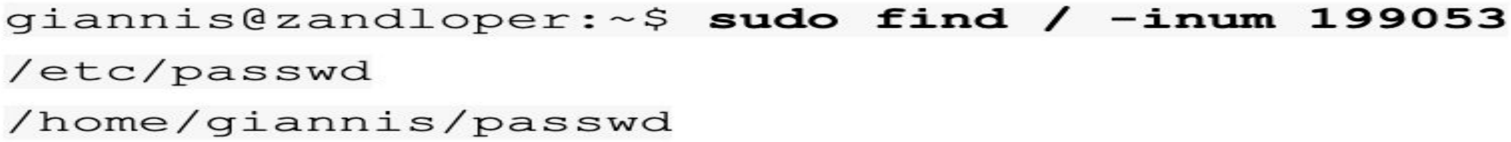
A hard link is actually nothing more than a regular directory entry, which in turn can be seen as a pointer to the actual file's data on the disk. The cool thing about hard-links is that a file can be stored once on the disk, and be linked to multiple times, from different locations/entries, without requiring to allocate extra disk space for each file instance.



```
gianni s@zandloper:/etc$ ls -i passwd
199053 passwd
```



```
giannis@zandloper:~$ ln /etc/passwd
giannis@zandloper:~$ ls -l passwd
-rw-r--r-- 2 root root 1402 2008-03-30 17:49 passwd
```



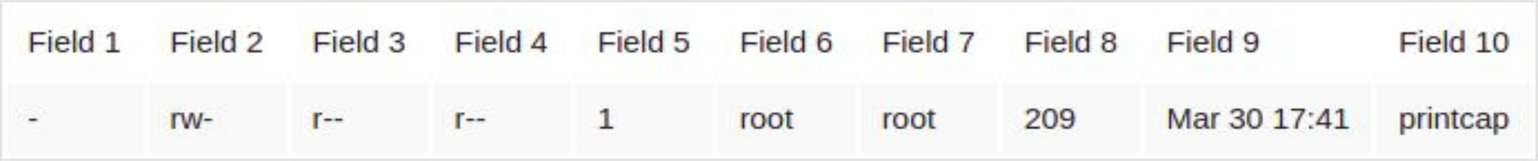
```
giannis@zandloper:~$ sudo find / -inum 199053
/etc/passwd
/home/giannis/passwd
```

UID and GID -- Associated with a user account

user ID (UID). $id -u; $echo $UID.

group ID (GID). $id

Every file/directory is associated with it’s owner’s (flield 6) and group’s ID (field 7).



Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10
-	rw-	r--	r--	1	root	root	209	Mar 30 17:41	printcap

v

Example: What happens when a user wants to open a file

- UID of user = *user1,*
- File to be written to = F,
- Owner of F = *user2*
- Permission triads with F = rwx r_ _ r_ _

To open F, *user1* needs to execute a program (called a *process)*

Example: $ vi F *(or $ gedit F)*

- Owner of vi = *user3*
- Permission triads of vi = rwx rw_ r_ _

Three new ids are associated with a process vi (I)

- RUID = Real user ID
- EUID = Effective User ID
- SUID = Saved User ID

These values depend on who runs vi, AND the first permission triad of P

Question 1: If *user1* runs vi and the first permission triad of vi is rwx then:

- RUID of vi = user1
- EUID of vi = user1
- SUID of vi = user1

Three new ids are associated with a process vi (II)

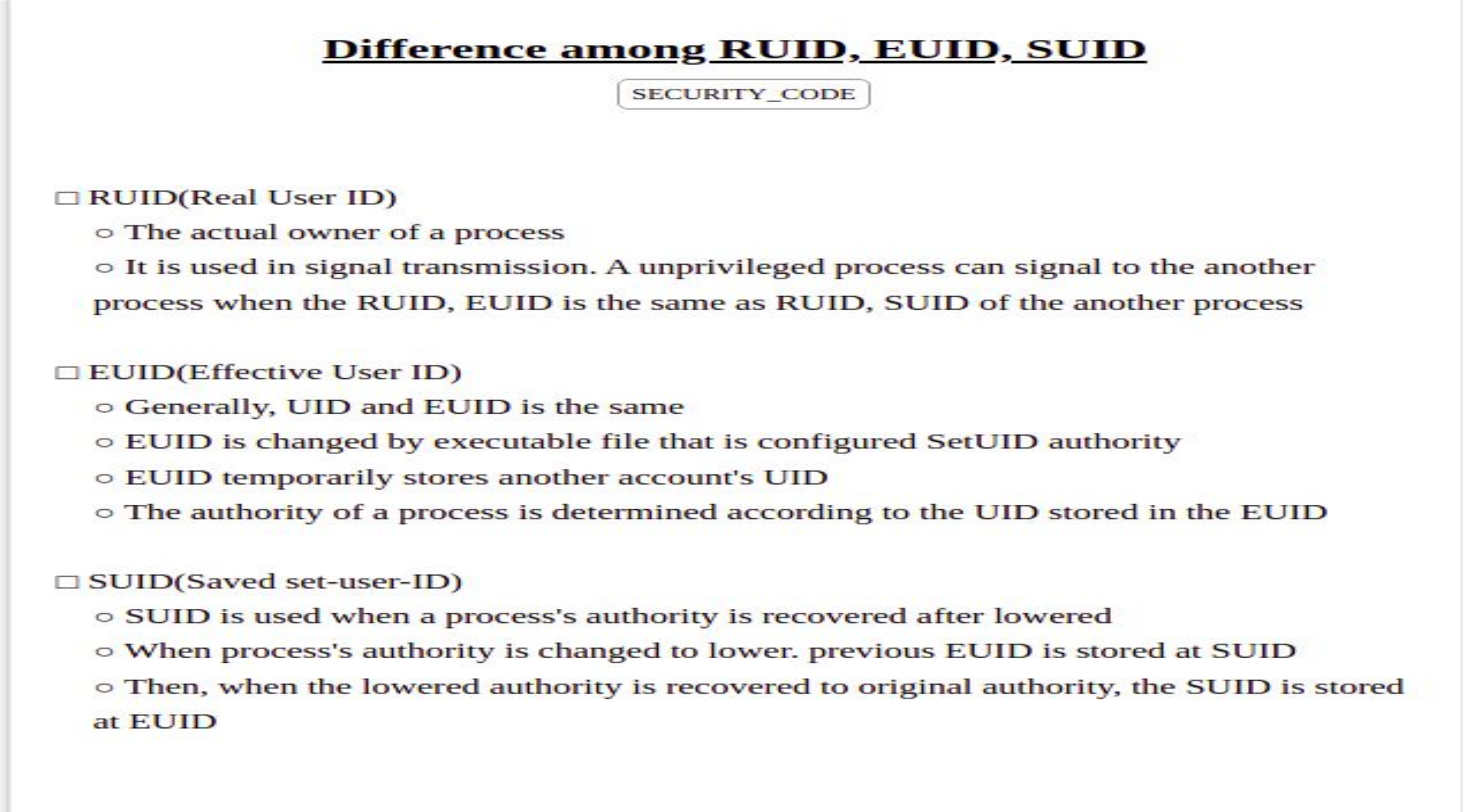
- RUID = Real user ID
- EUID = Effective User ID
- SUID = Saved User ID

These values depend on who runs vi, AND the first permission triad of P

Question 1: If *user1* runs vi and the first permission triad of vi is rws then:

- RUID of vi = user1
- EUID of vi = user3
- SUID of vi = user3

UID associated with a process: EUID, EGID, SUID and RUID



Difference among RUID, EUID, SUID

SECURITY_CODE

□ RUID(Real User ID)

- ○ The actual owner of a process
- ○ It is used in signal transmission. A unprivileged process can signal to the another process when the RUID, EUID is the same as RUID, SUID of the another process

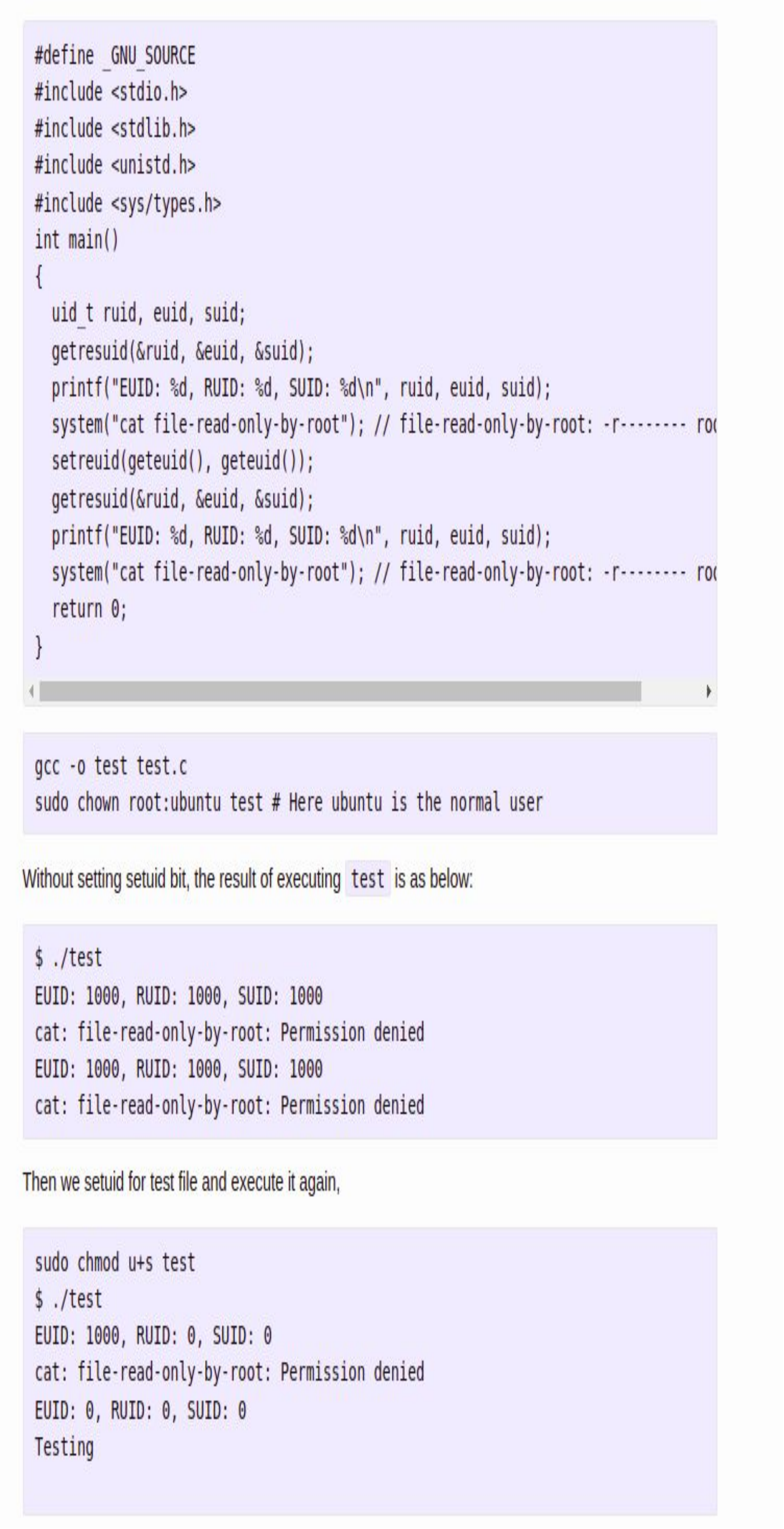
□ EUID(Effective User ID)

- ○ Generally, UID and EUID is the same
- ○ EUID is changed by executable file that is configured SetUID authority
- ○ EUID temporarily stores another account's UID
- ○ The authority of a process is determined according to the UID stored in the EUID

□ SUID(Saved set-user-ID)

- ○ SUID is used when a process's authority is recovered after lowered
- ○ When process's authority is changed to lower. previous EUID is stored at SUID
- ○ Then, when the lowered authority is recovered to original authority, the SUID is stored at EUID

Check EUID, RUID and SUID by executing the commands



```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
int main()
{
  uid_t ruid, euid, suid;
  getresuid(&ruid, &euid, &suid);
  printf("EUID: %d, RUID: %d, SUID: %d\n", ruid, euid, suid);
  system("cat file-read-only-by-root"); // file-read-only-by-root: -r-------- ro
  setreuid(geteuid(), geteuid());
  getresuid(&ruid, &euid, &suid);
  printf("EUID: %d, RUID: %d, SUID: %d\n", ruid, euid, suid);
  system("cat file-read-only-by-root"); // file-read-only-by-root: -r-------- ro
  return 0;
}
```

```
gcc -o test test.c
sudo chown root:ubuntu test # Here ubuntu is the normal user
```

Without setting setuid bit, the result of executing `test` is as below:

```
$ ./test
EUID: 1000, RUID: 1000, SUID: 1000
cat: file-read-only-by-root: Permission denied
EUID: 1000, RUID: 1000, SUID: 1000
cat: file-read-only-by-root: Permission denied
```

Then we setuid for test file and execute it again,

```
sudo chmod u+s test
$ ./test
EUID: 1000, RUID: 0, SUID: 0
cat: file-read-only-by-root: Permission denied
EUID: 0, RUID: 0, SUID: 0
Testing
```

Quiz 1:

*In this command “$vi F” executed by **user1**, will F be opened? Justify your answer.*

> *Clue:* *F will be opened if EUID of vi has the “open” privilege for F.*

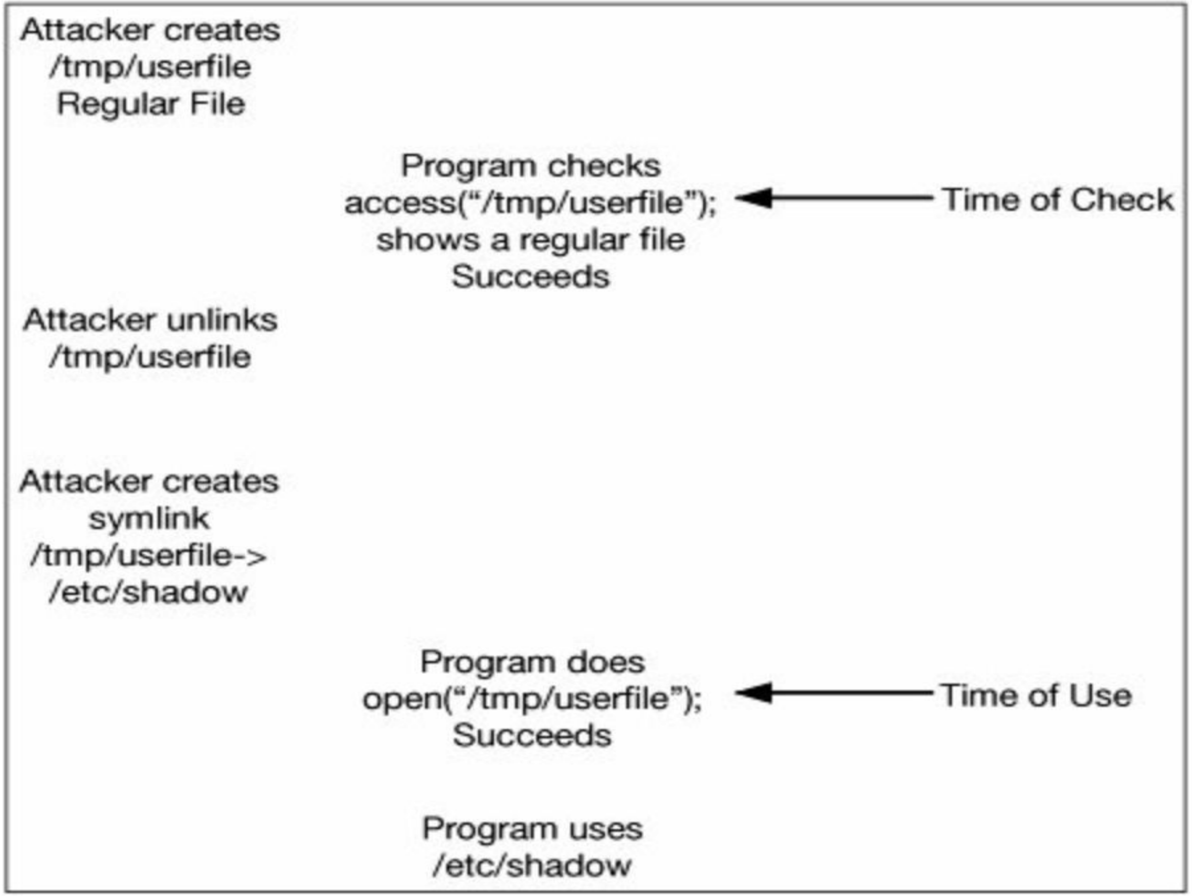
Quiz 2:

- All the passwords in a UNIX file system are stored in /etc/passwd
    - Owner of this file is root.
    - Permission triads of this file are: rwx r_ _ r_ _

- The passwd command (stored in /usr/bin/passwd) is used to write on /etc/passwd
    - Owner of this file is root.

How are all users able to write on /etc/passwd?

TOCTOU = Time-to-check and time-to-use (1)



Attacker creates /tmp/userfile Regular File

Program checks access("/tmp/userfile"); shows a regular file Succeeds ← Time of Check

Attacker unlinks /tmp/userfile

Attacker creates symlink /tmp/userfile-> /etc/shadow

Program does open("/tmp/userfile"); Succeeds ← Time of Use

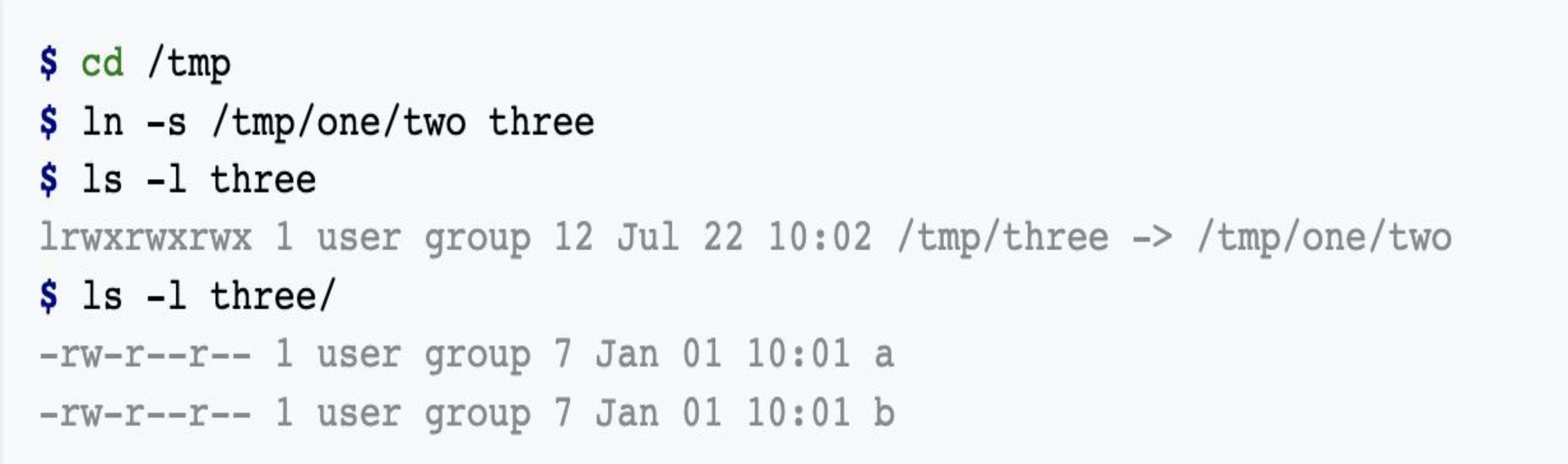
Program uses /etc/shadow

TOCTOU: content of /tmp/userfile is replaced with a symlink to /etc/passwd

```
res = access("/tmp/userfile", R_OK); //(1)

if (res!=0)

   die("access"); // (2)



   fd = open("/tmp/userfile", O_RDONLY);//(3)
```

Test.c with setuid bit set

Symlink



```
$ cd /tmp
$ ln -s /tmp/one/two three
$ ls -l three
lrwxrwxrwx 1 user group 12 Jul 22 10:02 /tmp/three -> /tmp/one/two
$ ls -l three/
-rw-r--r-- 1 user group 7 Jan 01 10:01 a
-rw-r--r-- 1 user group 7 Jan 01 10:01 b
```