Advanced Programming in the UNIX Environment

Week 03, Segment 2: UIDs and GIDs

Department of Computer Science
Stevens Institute of Technology

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

struct stat: st_mode, st_uid, st_gid

Every process has six or more IDs associated with it:

real user ID *real* group ID	who we really are
effective user ID *effective* group ID supplementary group IDs	used for file access permission checks
saved set-user-ID saved set-group-ID	saved by exec functions

Whenever a file is *setuid*, set the *effective* user ID to st_uid. Whenever a file is *setgid*, set the *effective* group ID to st_gid. (st_uid and st_gid always specify the owner and group owner of a file, regardless of whether it is setuid/setgid.)

```
apue$ ping -c 3 www.yahoo.com
PING new-fp-shed.wg1.b.yahoo.com (74.6.143.26): 56 data bytes
64 bytes from 74.6.143.26: icmp_seq=0 ttl=254 time=20.645093 ms
64 bytes from 74.6.143.26: icmp_seq=1 ttl=254 time=24.608150 ms
64 bytes from 74.6.143.26: icmp_seq=2 ttl=254 time=25.768367 ms

----new-fp-shed.wg1.b.yahoo.com PING Statistics----
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.645093/23.673870/25.768367/2.686381 ms
apue$ cp `which ping` .
apue$ ./ping -c 3 www.yahoo.com
ping: Cannot create socket: Operation not permitted
apue$ ls -l ping `which ping`
-r-sr-xr-x  1 root      wheel  40576 Feb 14  2020 /sbin/ping
-r-xr-xr-x  1 jschauma  users  40576 Sep 12 15:21 ping
apue$ chmod u+s ping
apue$ ls -l ping `which ping`
-r-sr-xr-x  1 root      wheel  40576 Feb 14  2020 /sbin/ping
-r-sr-xr-x  1 jschauma  users  40576 Sep 12 15:21 ping
apue$ ./ping -c 3 www.yahoo.com
ping: Cannot create socket: Operation not permitted
apue$
```


Terminal — 61×24
Hello, everybody!
Broadcast Message from fred@apue
(/dev/pts/1) at 15:40 UTC...
sup?
jschauma@apue$ ls -l /dev/pts
total 0
crw--w---- 1 jschauma tty 5, 0 Sep 12 15:41 0
crw--w---- 1 fred   tty 5, 1 Sep 12 15:40 1
jschauma@apue$ ls -l `which wall`
-r-xr-sr-x 1 root tty 26096 Feb 14 2020 /usr/bin/wall
jschauma@apue$
Terminal — 61×24
(/dev/pts/0) at 15:40 UTC...
Hello, everybody!
fred@apue$ wall
sup?
Broadcast Message from fred@apue
(/dev/pts/1) at 15:40 UTC...
sup?
fred@apue$

struct stat: st_mode, st_uid, st_gid

```
#include <unistd.h>

int setuid(uid_t uid);
int seteuid(uid_t uid);
```

Returns: 0 if OK, -1 on error

```
uid_t getuid(void);
uid_t geteuid(void);
```

Returns: uid_t; never errors

See also: getresuid(2) (if _GNU_SOURCE)

```
Terminal — 80×24
My effective uid is: 1000
My real uid is: 1001

Let's drop elevated privs and set our effective uid to our real uid.
Calling 'seteuid(1001)'...
My effective uid is: 1001
My real uid is: 1001

Because we used seteuid(1001) and not setuid(1001), we can get back our earlier
euid 1000.
Calling 'seteuid(1000)'...
My effective uid is: 1000
My real uid is: 1001

Now let's try to 'setuid(1001)'...
My effective uid is: 1001
My real uid is: 1001

Now let's try to gain back our effective uid 1000...
Calling 'seteuid(1000)'...
Unable to seteuid(1000): Operation not permitted
My effective uid is: 1001
My real uid is: 1001
fred@apue$
```

access(2)

```
#include <unistd.h>

int access(const char *path, int mode);
int faccessat(int fd, const char *path, int mode, int flags);
```

Returns: 0 if OK, -1 on error

Tests file accessibility on the basis of the *real* uid and gid. This allows setuid/setgid programs to see if the real user could access the file without it having to drop permissions to perform the check.

The mode parameter can be a bitwise OR of:

- `R_OK` – test for read permission
- `W_OK` – test for write permission
- `X_OK` – test for execute permission
- `F_OK` – test for existence of file

```
Terminal — 80×24
open ok for /etc/passwd
apue$ ls -l /etc/passwd
-rw-r--r--  1 root  wheel  1485 Sep 12 15:12 /etc/passwd
apue$ ./a.out /etc/master.passwd
access error for /etc/master.passwd
open error for /etc/master.passwd
apue$ ls -l /etc/master.passwd
-rw-------  1 root  wheel  1716 Sep 12 15:13 /etc/master.passwd
apue$ sudo chown root a.out
apue$ sudo chmod u+s a.out
apue$ ls -l a.out
-rwsr-xr-x  1 root  users  9536 Sep 12 17:04 a.out
apue$ ./a.out /etc/passwd
access ok for /etc/passwd
open ok for /etc/passwd
apue$ ./a.out /etc/master.passwd
access error for /etc/master.passwd
open ok for /etc/master.passwd
apue$ ls -l a.out
-rwsr-xr-x  1 root  users  9536 Sep 12 17:04 a.out
apue$ cc access.c
apue$ ls -l a.out
-rwxr-xr-x  1 jschauma  users  9536 Sep 12 17:05 a.out
apue$
```

All about UIDs

Each process has an *effective* UID and a *real* UID, just like an effective *GID* and a real *GID*.

If the setuid (setgid) bit is set in the permissions, the effective UID (GID) will become that of the `st_uid` (`st_gid`) of the file's `struct stat` at execution time.

You can switch between them via `seteuid(2)`; `setuid(2)` sets both irrevocably.

The effective UID and group ID are the ones used for file permission checks, but we can check whether the real ID would have permission via the access(2) system call.

Coming up next: file permission checks.