# Κεφάλαια 7 και 8 – RH124 Συμπληρωματικές σημειώσεις

**Γεώργιος Μαγκλάρας PhD**
Associate Instructor
RedHat Academy
Πανεπιστήμιο Δυτικής Αττικής

# Προγράμματα setuid setgid

- Ερήμην, χωρίς το set{uid,gid} όταν εκτελούμε ένα πρόγραμμα (στο οποίο έχουμε +x δικαιώματα), το πρόγραμμα εκτελείται με τα credentials του χρήστη που το εκτελεί.
- Με το set{uid,gid}, το πρόγραμμα θα εκτελεστεί με τα credentials του κατόχου χρήστη (owner) κατ της ομάδας χρήστη (group)

# Προγράμματα setuid setgid (2)

Για το αρχείο uid_demo_listdir που έχει τα εξής δικαιώματα:

**-rwxr-xr-x.  1 georgios georgios  17K Feb  9 14:03 uid_demo_listdir**

μπορούμε να προσθέσουμε το setuid,setgid bit με τις ακόλουθες εντολές:

- **chmod u+s uid_demo_listdir** (βάλε setuid)
- **chmod g+s uid_demo_listdir** (βάλε setgid)

  ή
- **chmod ug+s uid_demo_listdir** (βάλε και τα δύο με μια εντολή)
- **chmod 6755 uid_demo_listdir** (βάλε και τα δύο με οκταδική σημειολογία)

# Προγράμματα setuid setgid (3)

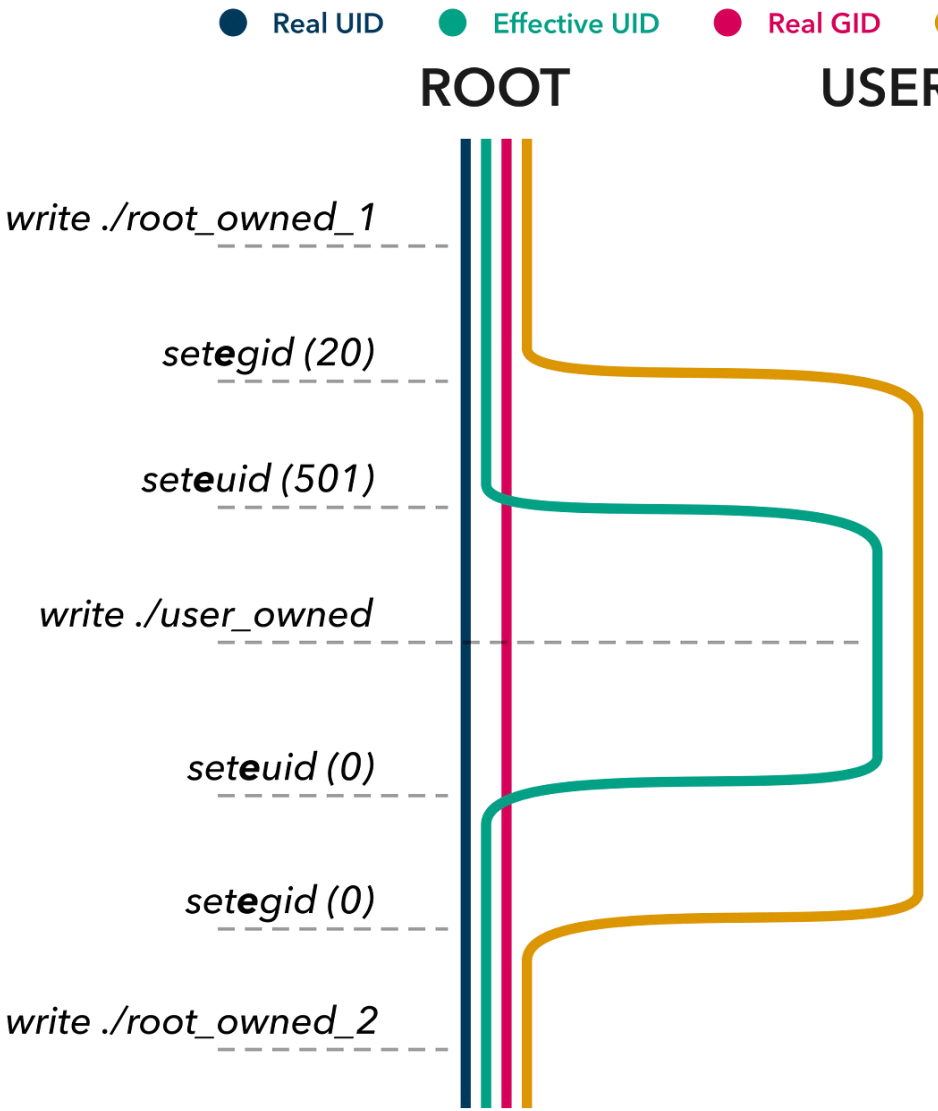
- Έτσι κατορθώνουν πολλές διαχειριστικές εντολές να εκτελούνται απο έναν απλό χρήστη (non root, no sudo) και να μπορούν να κάνουν τη δουλειά τους, τροποποιώντας αρχεία στα οποία μόνο ο διαχειριστής (root) θα είχε πρόσβαση.
- Παράδειγμα: Η εντολή **passwd**

# Προγράμματα setuid setgid (4)

- Με δεδομένο οτι η εντολή passwd έχει το setuid

  -rw**s**r-xr-x. 1 root root 90K Oct 15 02:00 /usr/bin/passwd

- Όταν ένας απλός τοπικός χρήστης την εκτελεί και αλλάξει τον κωδικό του (/etc/shadow), για μέρος της εκτέλεσής του, το πρόγραμμα εκτελεί με τα credentials του διαχειριστή (root) (privilege escallation → αναβάθμιση δικαιωμάτων) τροποποιεί το hash του χρήστη στο /etc/shadow και κατόπιν επιστρέφει στα non privileged credentials του χρήστη που το κάλεσε, με ασφάλεια (βλέπε επόμενο slide 6)

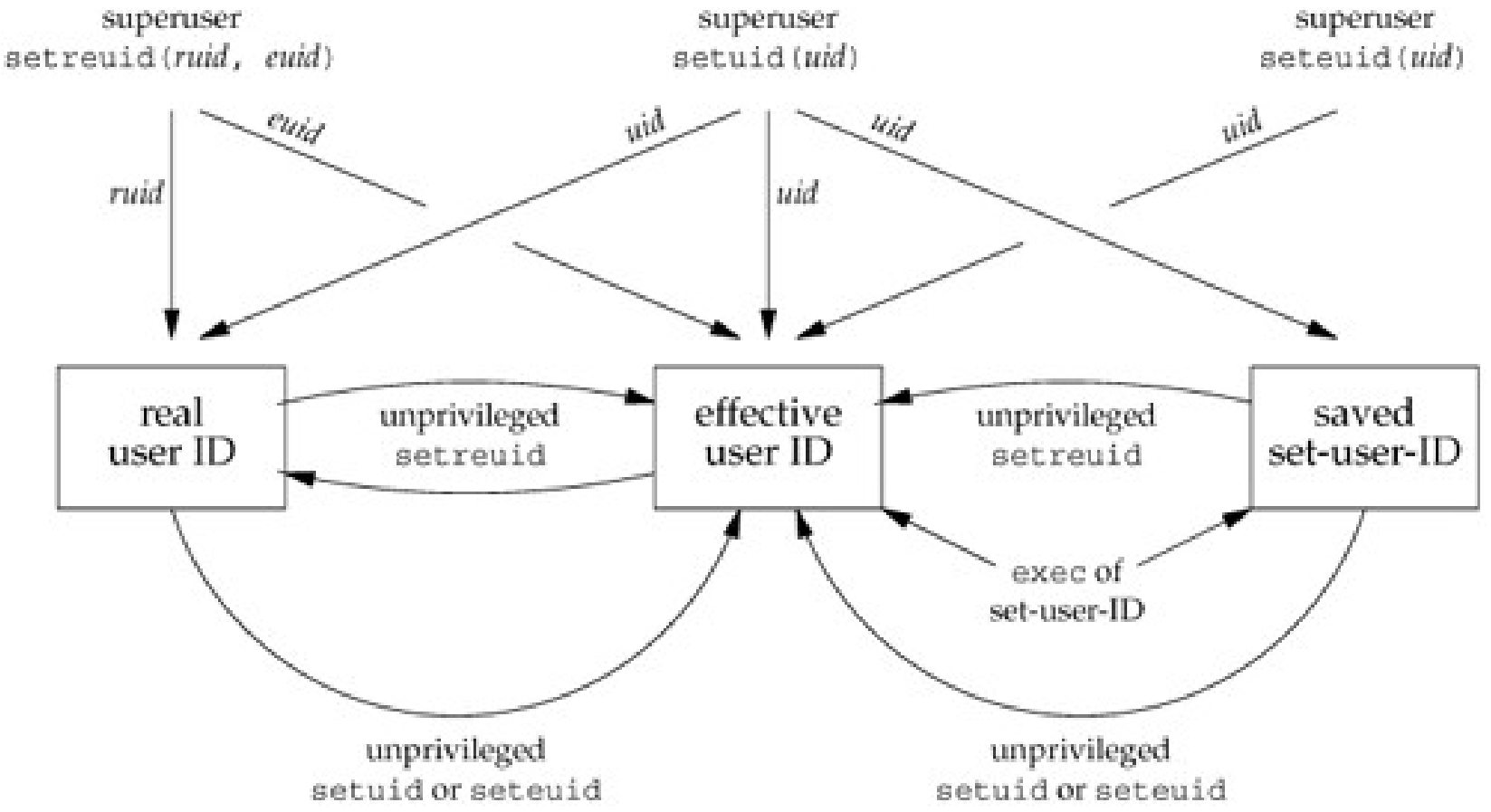



**Redhat Academy**
**Chapter 7**
**RH124 – 2025**

**How the passwd program works**

Georgios Magklaras

# Προγράμματα setuid setgid (6)



## 2.7 Processes

Put most simply, a *process* is an instance of an executing program. When a program is executed, the kernel loads the code of the program into virtual memory, allocates space for program variables, and sets up kernel bookkeeping data structures to record various information (such as process ID, termination status, user IDs, and group IDs) about the process.

From a kernel point of view, processes are the entities among which the kernel must share the various resources of the computer. For resources that are limited, such as memory, the kernel initially allocates some amount of the resource to the process, and adjusts this allocation over the lifetime of the process in response to the demands of the process and the overall system demand for that resource. When the process terminates, all such resources are released for reuse by other processes. Other resources, such as the CPU and network bandwidth, are renewable, but must be shared equitably among all processes.

### Process memory layout

A process is logically divided into the following parts, known as *segments*:

- *Text*: the instructions of the program.
- *Data*: the static variables used by the program.
- *Heap*: an area from which programs can dynamically allocate extra memory.
- *Stack*: a piece of memory that grows and shrinks as functions are called and return and that is used to allocate storage for local variables and function call linkage information.

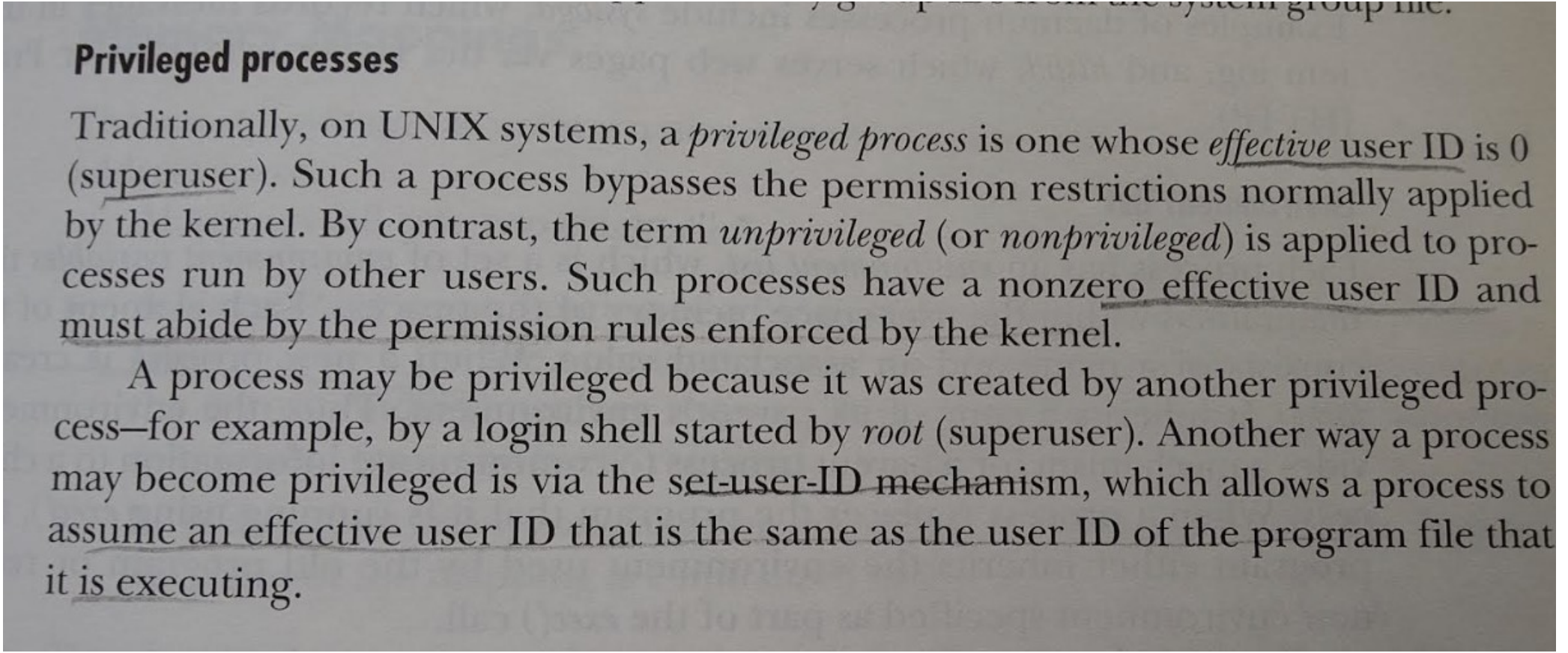
What is a process?

(applicable to chapters 7 and 8 of RH124)

Reference:
*The Linux Programming Interface Kerrisk M. ISBN-10: 1-59327-220-3*

What is a privileged process?

Reference:
*The Linux Programming Interface Kerrisk M. ISBN-10: 1-59327-220-3*



## Privileged processes

Traditionally, on UNIX systems, a *privileged process* is one whose *effective* user ID is 0 (superuser). Such a process bypasses the permission restrictions normally applied by the kernel. By contrast, the term *unprivileged* (or *nonprivileged*) is applied to processes run by other users. Such processes have a nonzero effective user ID and must abide by the permission rules enforced by the kernel.

A process may be privileged because it was created by another privileged process—for example, by a login shell started by *root* (superuser). Another way a process may become privileged is via the set-user-ID mechanism, which allows a process to assume an effective user ID that is the same as the user ID of the program file that it is executing.

# Πείραμα setuid/setgid

- Βήμα 1ο: Μεταγλώττιση προγράμματος σε C το οποίο επιχειρεί privilege escallation, χρησιμοποιώντας το μηχανισμό setuid (slides 11,12)
- Βήμα 2ο: Τρέξιμο του ίδιου προγράμματος απο μη σχετιζόμενο με τον ιδιοκτήτη του αρχείου χρήστη χωρίς δικαίωμα setuid. Επαλήθευση του ότι **δεν** έχουμε privilege escallation.
- Βήμα 3ο: Τρέξιμο του ίδιου προγράμματος απο τον ίδιο μη σχετιζόμενο με τον ιδιοκτήτη του αρχείου χρήστη ΜΕ δικαίωμα setuid. Επαλήθευση επιτυχούς privilege escallation.
- Συμπεράσματα

# Πείραμα setuid/setgid (2)

```
#include <stdio.h>
#include <unistd.h>
#include <dirent.h>

int main ()
{
    printf("real uid: %d\n", getuid());
    printf("effective uid: %d\n", geteuid());

    DIR *d;
    struct dirent *dir;
    d = opendir("/home/georgios");
    if (d) {
    while ((dir = readdir(d)) != NULL) {
      printf("%s\n", dir->d_name);
    }
    closedir(d);
  }
  return(0);
}
```

**Συνοπτική περιγραφή του πηγαίου κώδικα C:**

-Συμπεριέλαβε τις βιβλιοθήκες stdio, unistd και dirent

Στην κυρίως συνάρτηση (main): {

-Τύπωσε το real και effective uid του εκτελούμενου προγράμματος

-Προσπέλασε το home φάκελο του χρήστη georgios (***σημείωση:ο ιδιοκτήτης του αρχείου***)

-Τύπωσε τα περιοχόμενα του home φακέλου

}

# Πείραμα setuid/setgid (3)

Βήμα 1α)Σώσιμο του πηγαίου σε ένα αρχείο με όνομα **uid_demo_listdir.c ως χρήστης georgios** σε ένα path/φάκελο που όλοι οι χρήστες του συστήματος έχουν πρόσβαση (/tmp ή αλλού με δική σας επιλογή)

Βήμα 1β)Μεταγλώττιση του πηγαίου με τον GNU C Compiler ως χρήστης georgios :
**gcc -o uid_demo_listdir uid_demo_listdir.c**

Βήμα 1γ)Ελέγχουμε το παραχθέν εκτελέσιμο αρχείο για τύπο και δικαιώματα:
**file uid_demo_listdir**
uid_demo_listdir: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=e26595de15f98e346b21c3f193afb22915bb5b2c, for GNU/Linux 3.2.0, not stripped

**ls -l uid_demo_listdir**
**-rwxr-xr-x.** 1 georgios georgios 16936 Feb  9 14:03 uid_demo_listdir

# Πείραμα setuid/setgid (4)

Βήμα 2α)su – με την ταυτότητα ενός χρήστη (όχι του root) που δεν σχετίζεται με τον χρήστη **georgios** (αν δεν έχετε άλλο χρήστη στο σύστημα, δημιουργήστε έναν), λχ. **su - user1**

Βήμα 2β)Χωρίς uidset (βλέπε βήμα 1γ, slide 12) , τρέξτε ως χρήστης **user1** το εκτελέσιμο αρχείο:
**/tmp/uidtests/uid_demo_listdir**
real uid: 1002
effective uid: 1002

Βήμα 2γ)Επαληθεύστε οτι ΔΕΝ τυπώνονται τα περιεχόμενα του /home/georgios. Είναι δηλαδή σαν να πληκτρολογούσατε ‘ls -la /home/georgios’ ως χρήστης user1.

ΜΗ ΕΠΑΛΗΘΕΥΣΙΜΟ Privilege escallation.

# Πείραμα setuid/setgid (5)

Βήμα 3α)Ως χρήστης **georgios**, προσθέστε το setuid bit του εκτελέσιμου αρχείου και επαληθεύσατε οτι το κάνατε σωστά:
**chmod 4755 uid_demo_listdir**
**ls -l uid_demo_listdir**
-rw**s**r-xr-x. 1 georgios georgios 16936 Feb  9 14:03 uid_demo_listdir

Βήμα 3β)Με uidset, ξανατρέξτε ως χρήστης **user1** το εκτελέσιμο αρχείο και παρατηρείστε τη διαφορά:
**/tmp/uidtests/uid_demo_listdir**
real uid: **1002**
effective uid: **1000** →**(uid of georgios)**
.
..
.bash_logout

Βήμα 3γ)Επαληθεύστε οτι τυπώνονται τα περιεχόμενα του /home/georgios και άρα κατορθώσατε να έχετε privilege escallation!

# Πείραμα setuid/setgid (6)

## ΣΥΜΠΕΡΑΣΜΑΤΑ

-Οι μηχανισμοί setuid/setgid είναι χρήσιμοι όταν έχουμε καλώς ελεγχόμενες διαδικασίες αναβάθμισης δικαιωμάτων (privilege escallation). Το ‘καλώς ελεγχόμενες’ σημαίνει οτι τα προγράμματα που χρησιμοποιούν αυτά τα δικαιώματα και θέτουν το effective uid πρέπει να είναι σωστά γραμμένα, να έχουν καλώς ορισμένο σκοπό και να μην αφήνουν περιθώρια σε τρίτους να εκμεταλλευτούν κακόβουλα αυτή τη διαδικασία.


-Τα δικαιώματα setuid/setgid πρέπει να αποδίδονται με προσοχή, ειδικότερα σε προγράμματα των οποίων τον πηγαίο κώδικα δεν γνωρίζουμε! Ένα σημαντικό μέρος των επιθέσεων privilege escallation οφείλεται σε κακογραμμένο κώδικα που δεν διαχειρίζεται καλά αυτά τα δικαιώματα.

-Οι καλοί διαχειριστές συστημάτων πάντοτε σκανάρουν για την ύπαρξη τέτοιων αρχείων:
**find /vm2/uidtests/ ! -fstype proc -perm "4755" -print | xargs -n 50 sha256sum**

# Sticky bit

- Αρχικά ένας μηχανισμός για να κρατούνται αρχεία στην κυρίως μνήμη RAM και να αποφεύγεται το swapping.
- Σήμερα χρησιμοποιείται για να δίνει σε αρχεία και φακέλους τη δυνατότητα να μην αφαιρούνται ή/και να μετονομάζονται απο οποιονδήποτε άλλο χρήστη εκτός απο τον κάτοχό τους και τον διαχειριστή. Αυτό συμβαίνει και αν οι άλλοι χρήστες έχουν δικαιώματα (+w,x απο group (g) και others (o) ) για να το πράξουν.
- Για παράδειγμα, τα δικαιώματα του φακέλου /tmp:

  **drwxrwxrwt.  75 root     root 1.7K Feb  9 19:27 tmp**

# Sticky bit (2)

- **mkdir stickydir**
- **chmod +1777 stickydir**

  Θα μας δώσει ένα φάκελο:

  drwxrwxrw**t**.  2 georgios georgios     6 Feb  9 19:32 stickydir
- touch mystickyfile.txt
- chmod 1777 mystickyfile.txt

  Θα μας δώσει ένα αρχείο όπως το παρακάτω:

  -rwxrwxrw**t**.  1 georgios georgios    0 Feb  9 19:37 mystickyfile.txt

# Umask - ερήμην δικαιώματα

- Όταν δημιουργούμε ένα αρχείο ή φάκελο, η δημιουργία τους γίνονται πάντοτε με ερήμην δικαιώματα.
- Αυτά καθορίζονται απο την εντολή umask

  **umask**

  0022

# Umask- ερήμην δικαιώματα (2)

- Για να καταλάβουμε πως προκαθορίζει δικαιώματα η umask έχουμε δύο τρόπους:
- Α)Χρησιμοποιώντας την οκταδική σημειολογία να αφαιρέσουμε απο τα πλήρη δικαιώματα την τιμή του umask:

  0777 (πλήρη δικαιώματα)

  -0022 (επιλεγμένη τιμή umask)

  ____

  0755

- Δηλαδή μια umask 0022 δημιουργεί ερήμην:
  - Αρχεία πλήρως προσπελάσιμια απο τον κάτοχο, μόνο για ανάγνωση και εκτέλεση απο την ομάδα και μόνο για ανάγνωση και εκτέλεση απο όλους τους άλλους
  - Φακέλους πλήρως προσπελάσιμους απο τον κάτοχο, και πρόσβαση ανάγνωσης μόνο για ομάδα και άλλους.

# Umask- ερήμην δικαιώματα (3)

- Β)Χρησιμοποιώντας το συμπλήρωμα ως προς 1 ενός δυαδικού αριθμού ως εξής:

| | Οκταδικό | Δυαδικό |
|---|---|---|
| **Πλήρης δικαιώματα** | 0777 | 000 111 111 111 |
| **Επιλεγμένη umask** | 0022 | 000 000 010 010 |
| **Συμπλήρωμα ως προς 1 της umask** | ----- | 111 111 101 101 |
| **Παραχθέν δικαίωμα** | **0755** | **000 111 101 101** |