Linux Security and Isolation APIs Control Groups (cgroups)

Michael Kerrisk, man7.org © 2020

mtk@man7.org

February 2020

Outline

13	Cgroups	13-1
13.1	Introduction to cgroups v1 and v2	13-3
13.2	Cgroups v1: hierarchies and controllers	13-17
13.3	Cgroups v1: populating a cgroup	13-24
13.4	Cgroups v1: release notification	13-33
13.5	Cgroups v1: a survey of the controllers	13-43
13.6	Cgroups /proc files	13-65
13.7	Cgroup namespaces	13-68

Outline

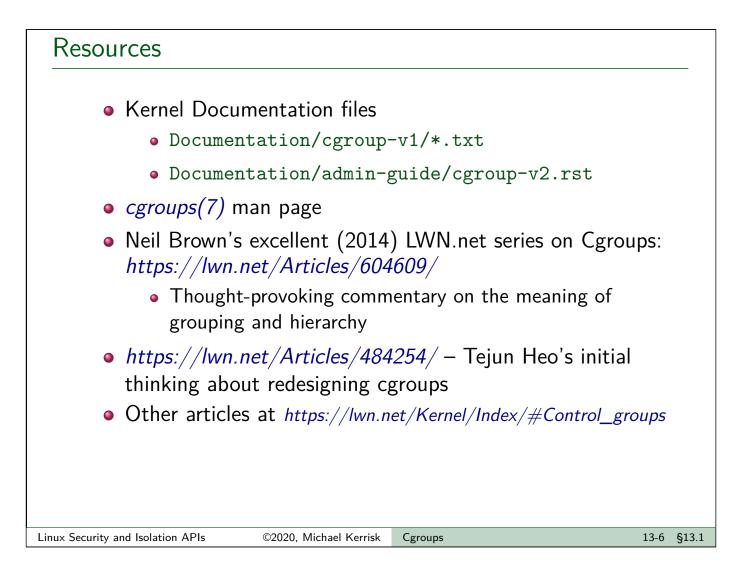
13	Cgroups	13-1
13.1	Introduction to cgroups v1 and v2	13-3
13.2	Cgroups v1: hierarchies and controllers	13-17
13.3	Cgroups v1: populating a cgroup	13-24
13.4	Cgroups v1: release notification	13-33
13.5	Cgroups v1: a survey of the controllers	13-43
13.6	Cgroups /proc files	13-65
13.7	Cgroup namespaces	13-68

Goals

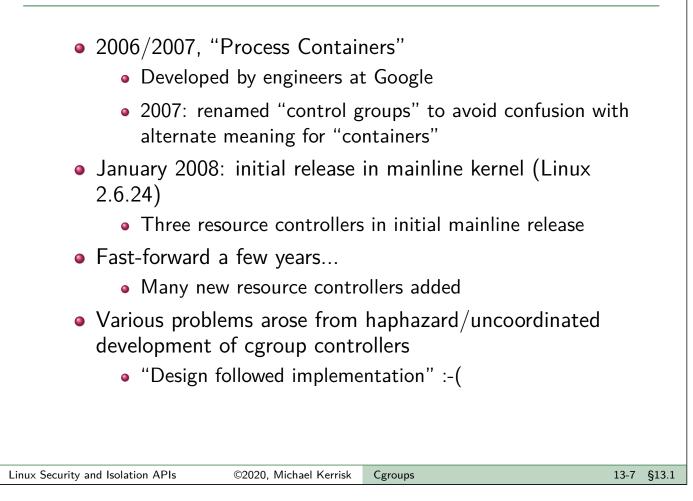
- Cgroups is a big topic
 - Many controllers
 - V1 versus V2 interfaces
- Our goal: understand fundamental semantics of cgroup filesystem and interfaces
 - Useful from a programming perspective
 - How do I build container frameworks?
 - What else can I build with cgroups?
 - And useful from a system engineering perspective
 - What's going on underneath my container's hood?

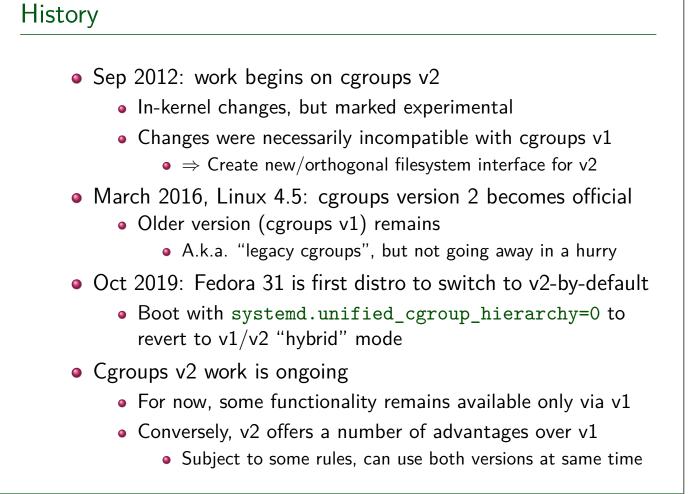
Focus

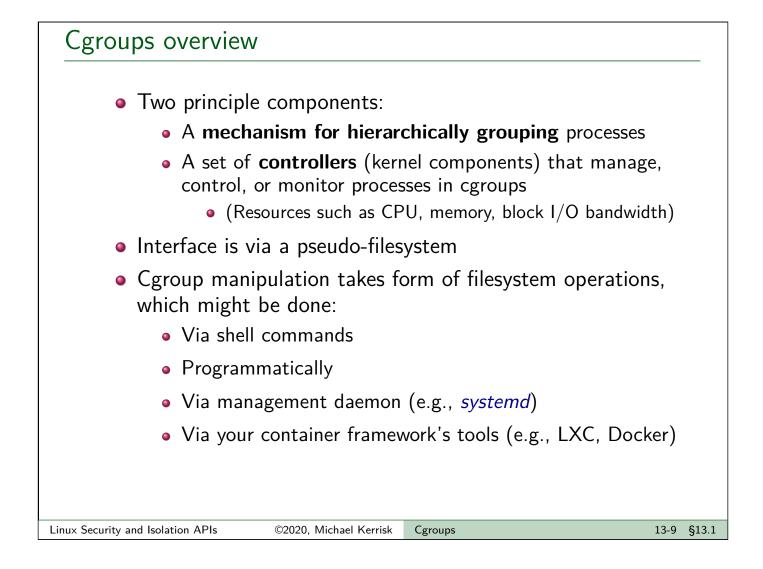
We'll focus on:
General principles of operation; goals of cgroups
The cgroup filesystem
Interacting with the cgroup filesystem using shell commands
Problems with cgroups v1, motivations for cgroups v2
Differences between cgroups v1 and v2
We'll look briefly at some of the controllers



History

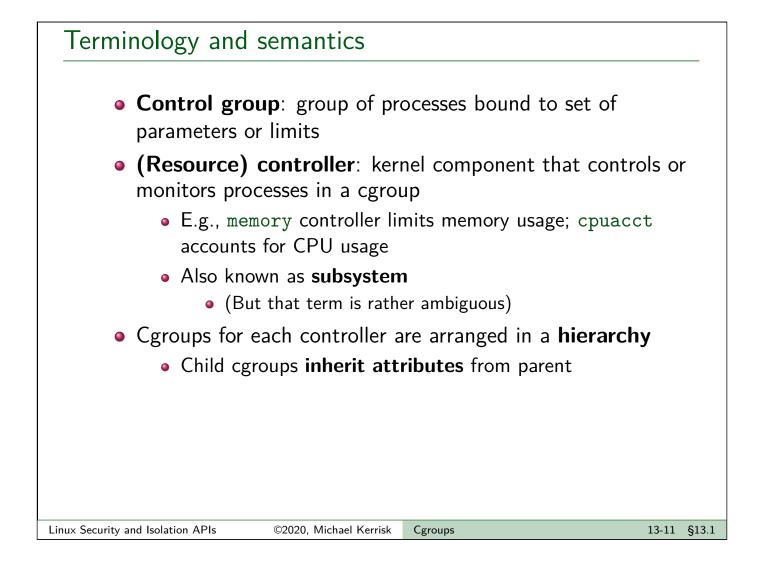


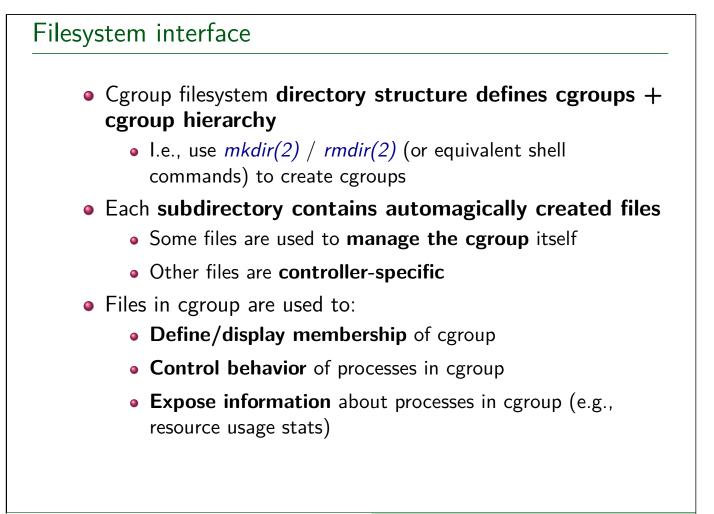


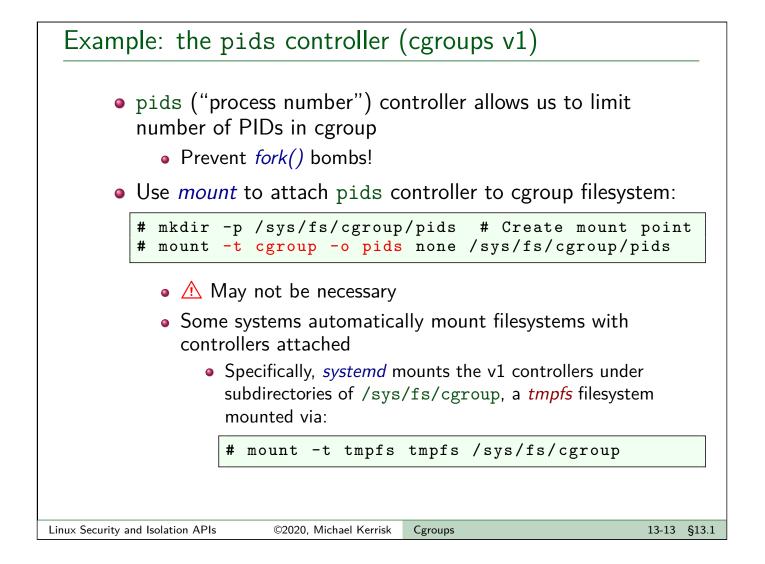


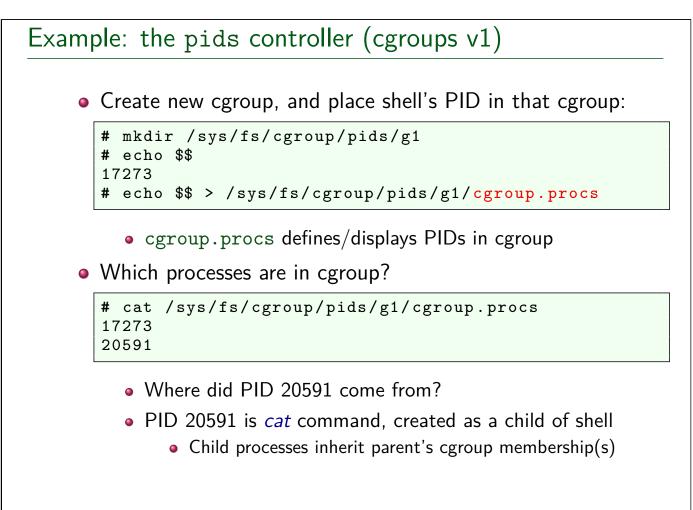
What do cgroups allow us to do?

- Limit resource usage of group
 - E.g., limit percentage of CPU available to group
- Prioritize group for resource allocation
 - E.g., some group might get greater proportion of CPU
- Resource accounting
 - Measure resources used by processes
- Freeze a group
 - Freeze, restore, and checkpoint a group
- And more...

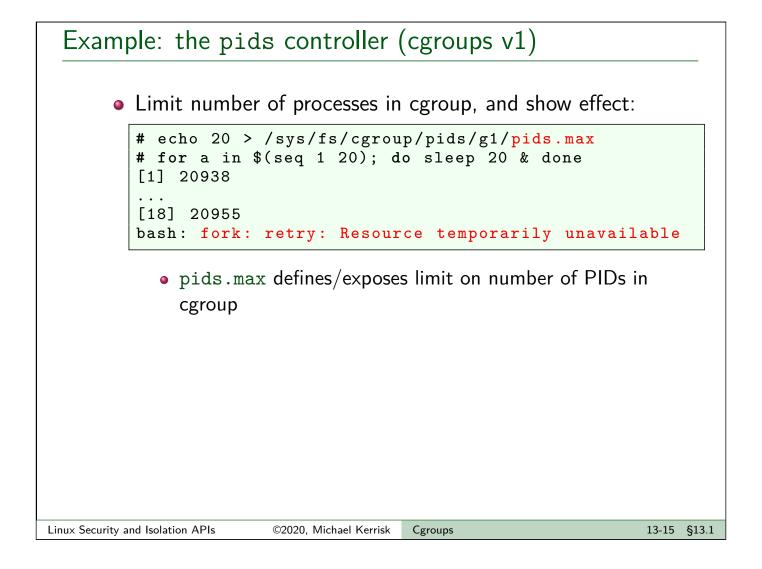


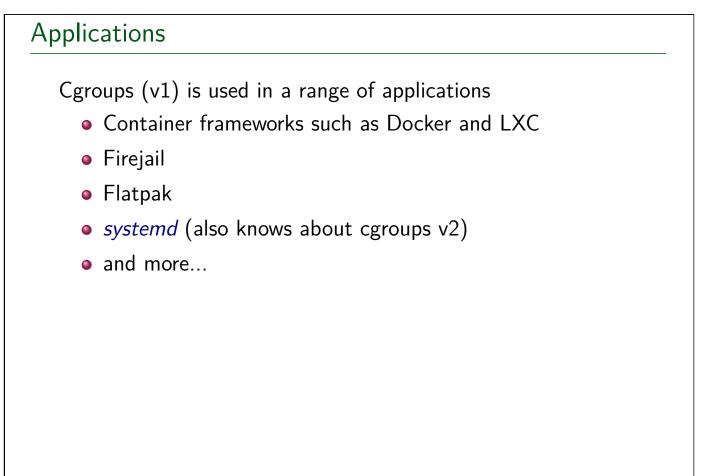






Linux Security and Isolation APIs



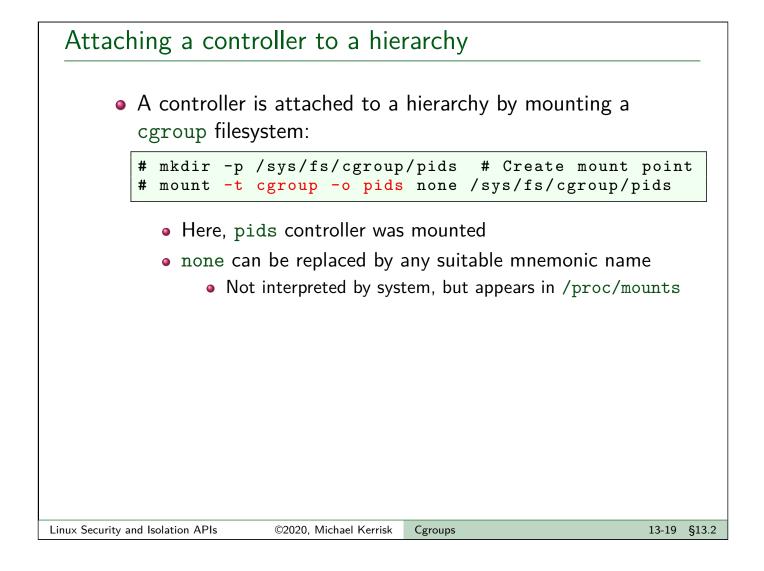


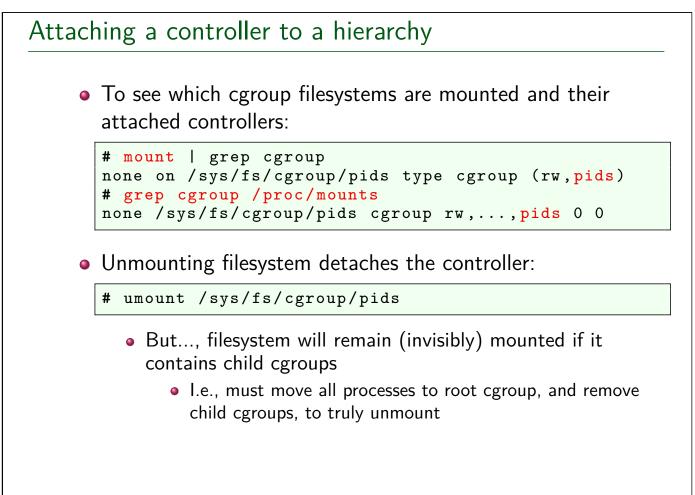
Outline

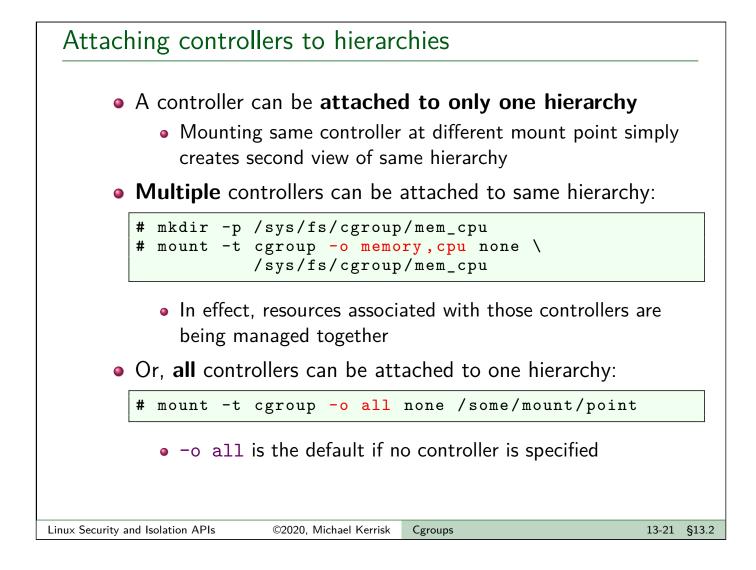
13	Cgroups	13-1
13.1	Introduction to cgroups v1 and v2	13-3
13.2	Cgroups v1: hierarchies and controllers	13-17
13.3	Cgroups v1: populating a cgroup	13-24
13.4	Cgroups v1: release notification	13-33
13.5	Cgroups v1: a survey of the controllers	13-43
13.6	Cgroups /proc files	13-65
13.7	Cgroup namespaces	13-68

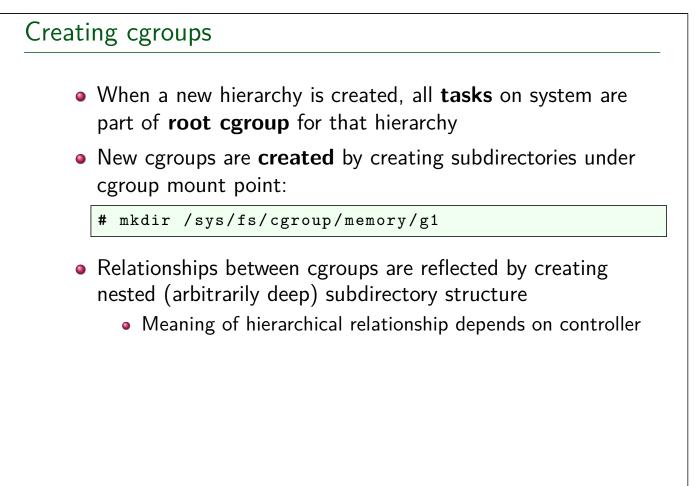
Cgroup hierarchies

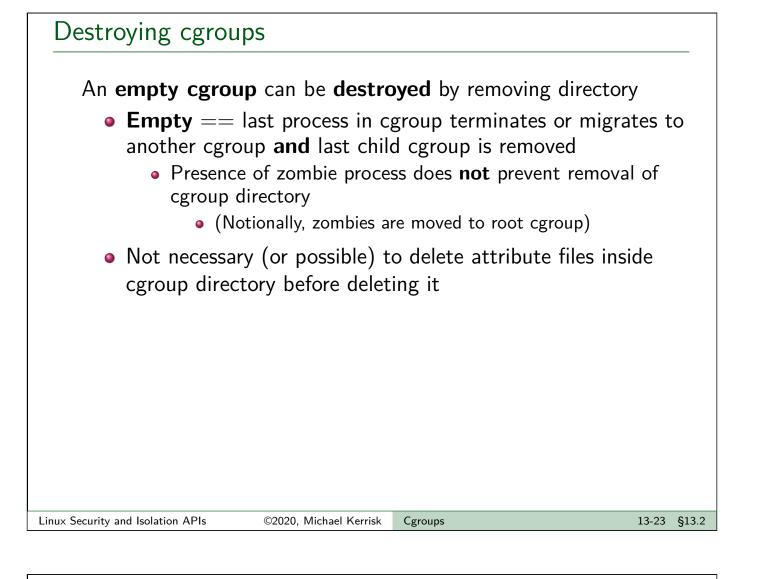
- **Cgroup** == collection of processes
- **Cgroup hierarchy** == hierarchical arrangement of cgroups
 - Implemented via a cgroup pseudo-filesystem
- Structure and membership of cgroup hierarchy is defined by:
 - Mounting a cgroup filesystem
 - Oreating a subdirectory structure that reflects desired cgroup hierarchy
 - Oving processes within hierarchy by writing their PIDs to special files in cgroup subdirectories
 - E.g., cgroup.procs











Outline

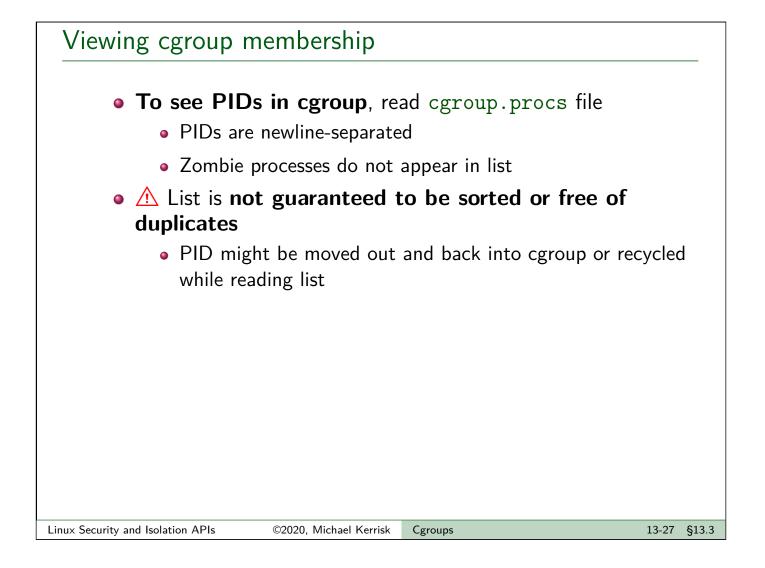
13	Cgroups	13-1
13.1	Introduction to cgroups v1 and v2	13-3
13.2	Cgroups v1: hierarchies and controllers	13-17
13.3	Cgroups v1: populating a cgroup	13-24
13.4	Cgroups v1: release notification	13-33
13.5	Cgroups v1: a survey of the controllers	13-43
13.6	Cgroups /proc files	13-65
13.7	Cgroup namespaces	13-68

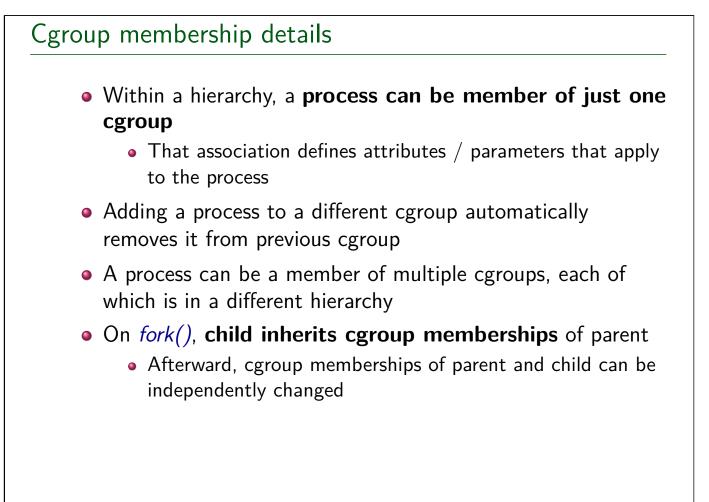
Placing a process in a cgroup

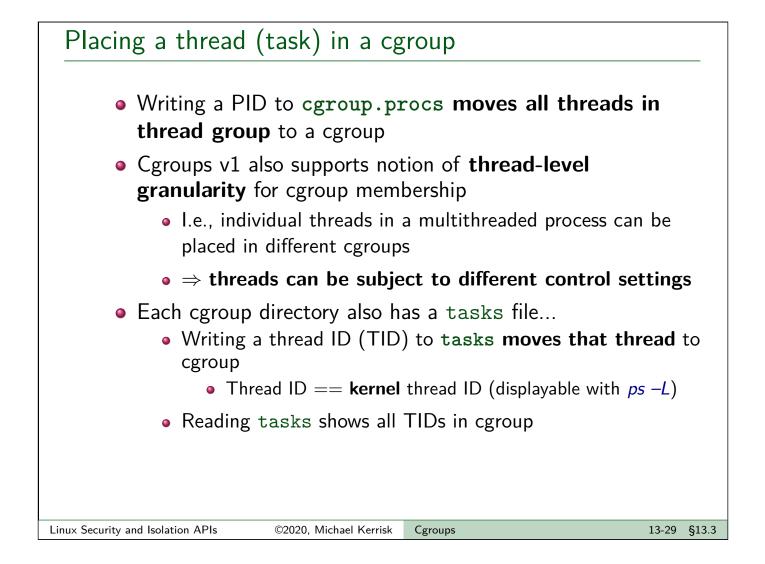
• To move a **process** to a cgroup, we write its PID to cgroup.procs file in corresponding subdirectory

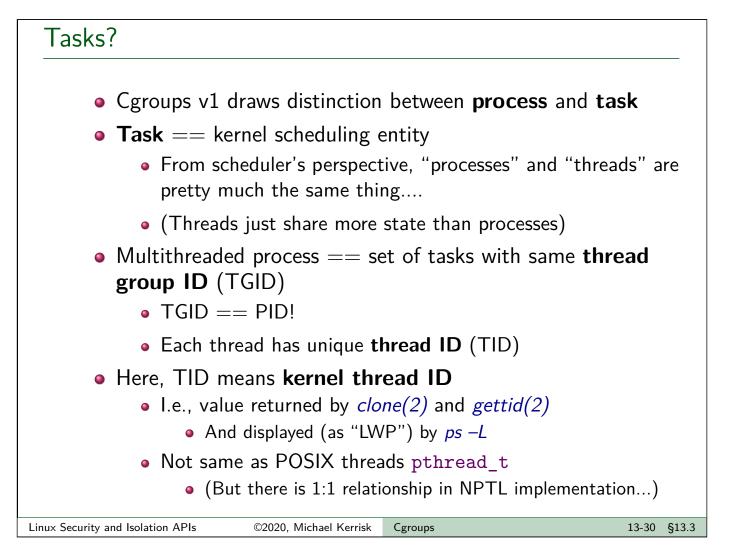
echo \$\$ > /sys/fs/cgroup/memory/g1/cgroup.procs

- In multithreaded process, moves all threads to cgroup...
- Can write only one PID at a time
 - write() fails with EINVAL
- Writing 0 to cgroup.procs moves writing process to cgroup









Exercises			
		s to cgroups v2 only, reboot v y=0 to revert to "hybrid" mod	
-	cise, we create a cgrou te that process to a dif	p, place a process in the cgro ferent cgroup.	up, and
• If the	memory cgroup is not	already mounted, mount it:	
# mkd # mou	p 'cgroup.*mem' /pro ir -p /sys/fs/cgroup nt -t cgroup -o memo /sys/fs/cgroup/memor	/memory ory none /sys/fs/cgroup/mem	
	patched kernel that dis default. If you can't m necessary to reboot wi	e.g., some Debian releases) pro- sables the memory controller b ount the controller, it may be th the cgroup_enable=memor ption. Alternatively, you could this exercise.	y ry
	[Exercise continues o	n the next slide]	
Linux Security and Isolation APIs	©2020, Michael Kerrisk	Cgroups	13-31 §13.3

 Create two subdirectories, m1 and m2, in the memory cgroup root directory.
 Execute the following command, and note the PID assigned to the resulting process:
sleep 300 &
 Write the PID of the process created in the previous step into the file m1/cgroup.procs, and verify by reading the file contents.
• Now write the PID of the process into the file m2/cgroup.procs
Is the PID still visible in the file m1/cgroup.procs? Explain.
 Try removing cgroup m1 using the command rm -rf m1. Why doesn't this work?
• Remove the cgroups m1 and m2 using the <i>rmdir</i> command.