Linux Security and Isolation APIs Fundamentals

Control Groups (cgroups): Introduction

Michael Kerrisk, man7.org © 2025

January 2025

mtk@man7.org

Outline	Rev: #8d7fc39ab521
13 Cgroups: Introduction	13-1
13.1 Preamble	13-3
13.2 What are control groups?	13-6
13.3 An example: the pids controller	13-13
13.4 Creating, destroying, and populating a cgroup	13-17
13.5 Exercises	13-24
13.6 Enabling and disabling controllers	13-31
13.7 Exercises	13-46

Outline **Cgroups:** Introduction 13 13-1 13.1 Preamble 13 - 313.2 What are control groups? 13 - 613.3 An example: the pids controller 13 - 1313.4 Creating, destroying, and populating a cgroup 13-17 13.5 Exercises 13 - 2413.6 Enabling and disabling controllers 13-31 13.7 Exercises 13 - 46

Goals

• We'll focus on:

- General principles of operation; goals of cgroups
- The cgroup2 filesystem
- Interacting with cgroup2 filesystem using shell commands
 - By 2021, all major distros switched to cgroups v2, so we'll ignore cgroups v1
- We'll look briefly at some of the controllers

Resources

- Kernel documentation files
 - V2: Documentation/admin-guide/cgroup-v2.rst
 - V1: Documentation/admin-guide/cgroup-v1/*.rst
 - Before Linux 5.3: Documentation/cgroup-v1/*.txt
- cgroups(7) manual page
- Chris Down, 7 years of cgroup v2, https://www.youtube.com/watch?v=LX6fM1IYZcg
- Neil Brown's (2014) LWN.net series on cgroups: https://lwn.net/Articles/604609/
 - Thought-provoking ideas on the meaning of grouping & hierarchy
- https://lwn.net/Articles/484254/ Tejun Heo's initial thoughts about redesigning cgroups (Feb 2012)
 - See also https://lwn.net/Articles/484251/, *Fixing Control Groups*, Jon Corbet, Feb 2012
- Other articles at https://lwn.net/Kernel/Index/#Control_groups

Security and Isolation APIs Fundamentals©2025 M. Kerrisk

Cgroups: Introduction

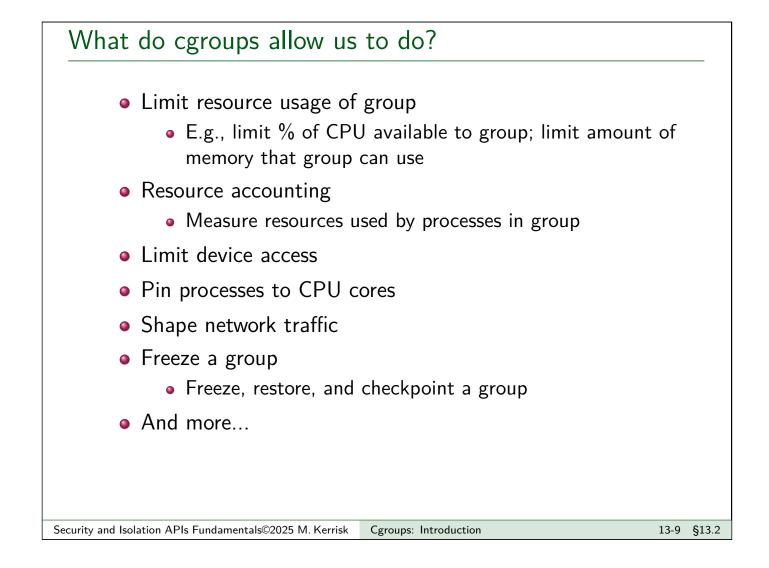
13-5 §13.1

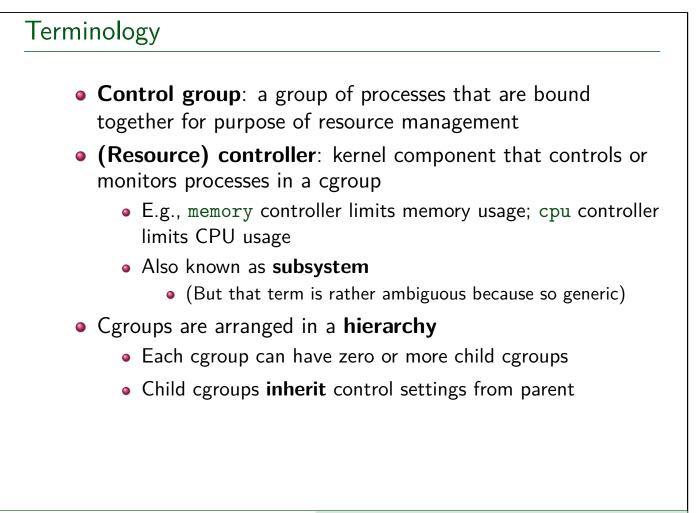
Outline

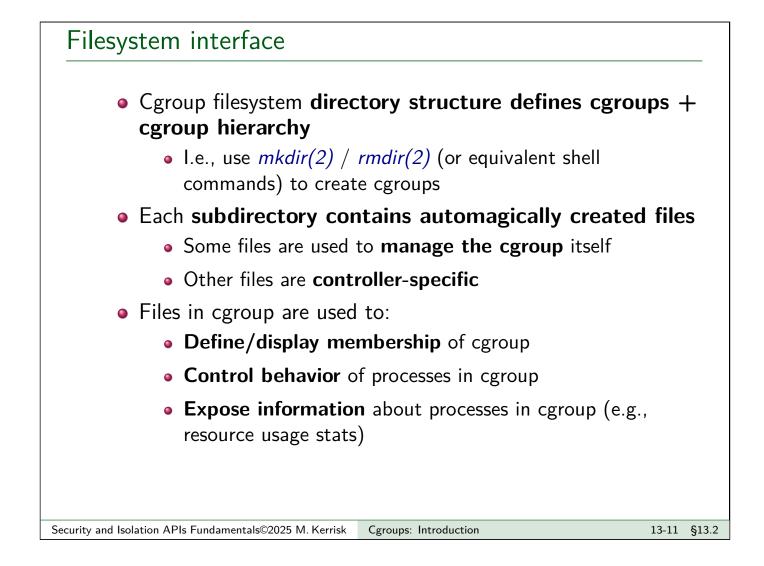
13 Cgroups: Introduction	13-1
13.1 Preamble	13-3
13.2 What are control groups?	13-6
13.3 An example: the pids controller	13-13
13.4 Creating, destroying, and populating a cgroup	13-17
13.5 Exercises	13-24
13.6 Enabling and disabling controllers	13-31
13.7 Exercises	13-46

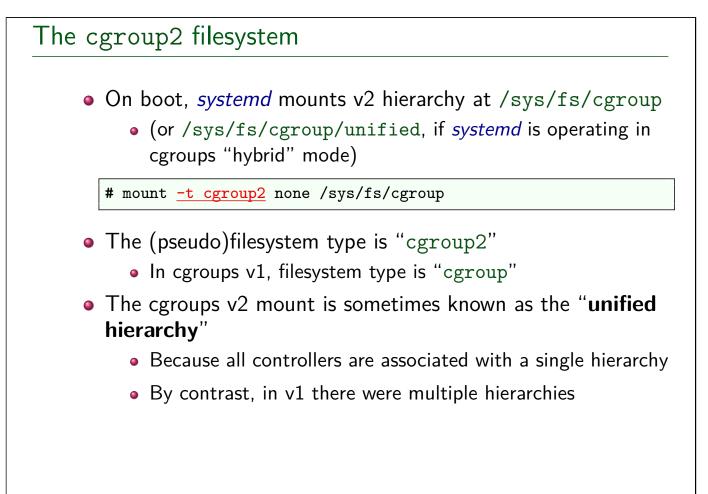
What are control groups?

- Two principal components:
 - A mechanism for hierarchically grouping processes
 - A set of **controllers** (kernel components) that manage, control, or monitor processes in cgroups
- Interface is via a pseudo-filesystem
- Cgroup manipulation takes form of filesystem operations, which might be done:
 - Via shell commands
 - Programmatically
 - Via management daemon (e.g., *systemd*)
 - Via your container framework's tools (e.g., LXC, Docker)









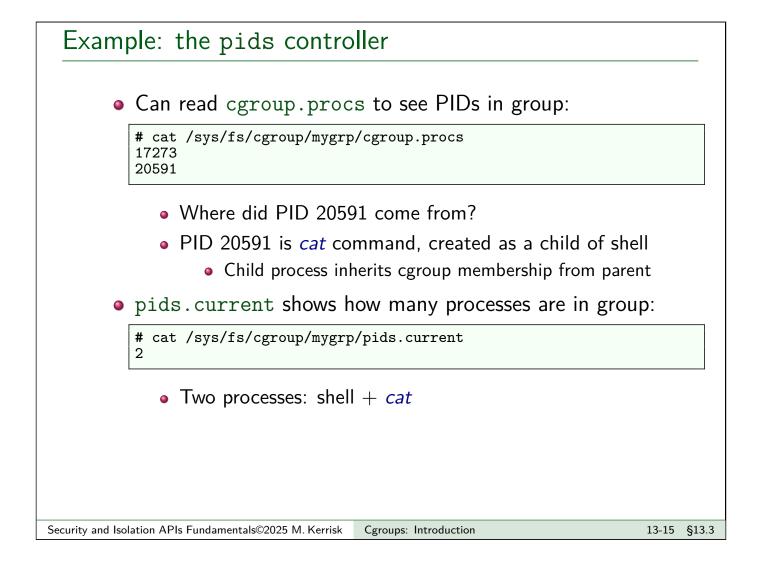
Outline Cgroups: Introduction 13 13-1 13.1 Preamble 13 - 313.2 What are control groups? 13 - 613.3 An example: the pids controller 13 - 1313.4 Creating, destroying, and populating a cgroup 13-17 13.5 Exercises 13 - 2413.6 Enabling and disabling controllers 13 - 3113.7 Exercises 13 - 46

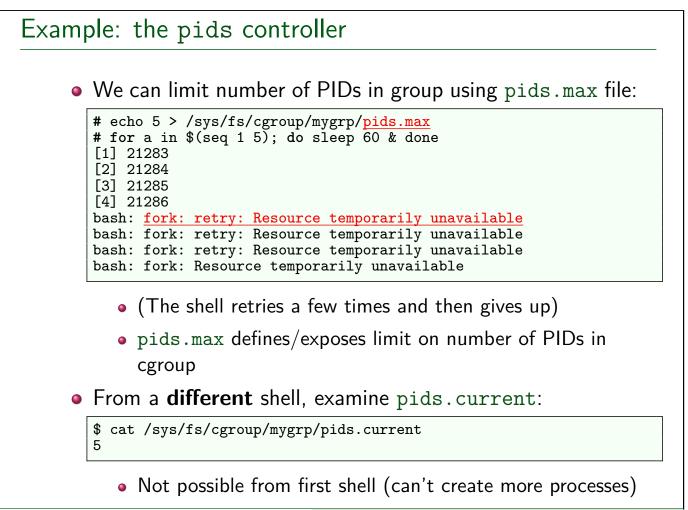
Example: the pids controller

- pids ("process number") controller allows us to limit number of PIDs in cgroup (prevent *fork()* bombs!)
- Create new cgroup, and place shell's PID in that cgroup:

```
# mkdir /sys/fs/cgroup/mygrp
# echo $$
17273
# echo $$ > /sys/fs/cgroup/mygrp/cgroup.procs
```

- cgroup.procs defines/displays PIDs in cgroup
- (Note '#' prompt \Rightarrow all commands done as superuser)
- Moving a PID into a group automatically removes it from group of which it was formerly a member
 - I.e., a process is always a member of exactly one group in the hierarchy





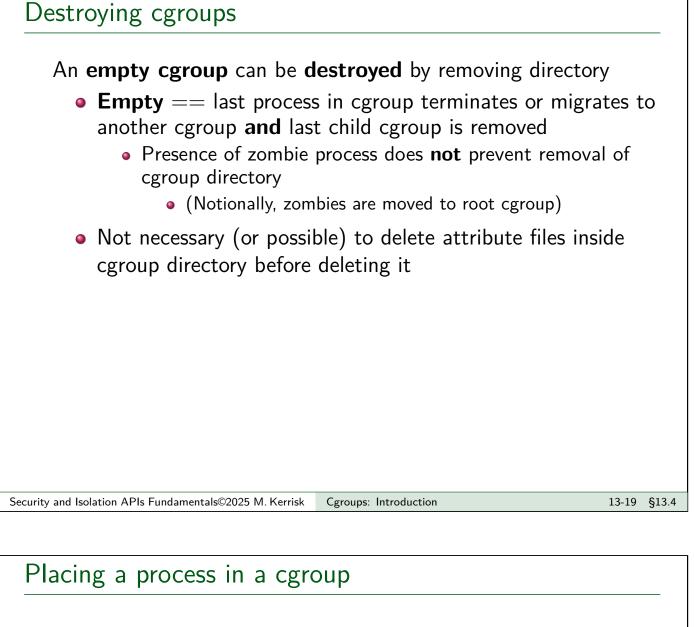
Outline Cgroups: Introduction 13 13-1 13.1 Preamble 13 - 313.2 What are control groups? 13 - 613.3 An example: the pids controller 13 - 1313.4 Creating, destroying, and populating a cgroup 13-17 13.5 Exercises 13 - 2413.6 Enabling and disabling controllers 13-31 13.7 Exercises 13 - 46

Creating cgroups

- Initially, all processes on system are members of root cgroup
- New cgroups are **created** by creating subdirectories under cgroup mount point:

```
# mkdir /sys/fs/cgroup/mygrp
```

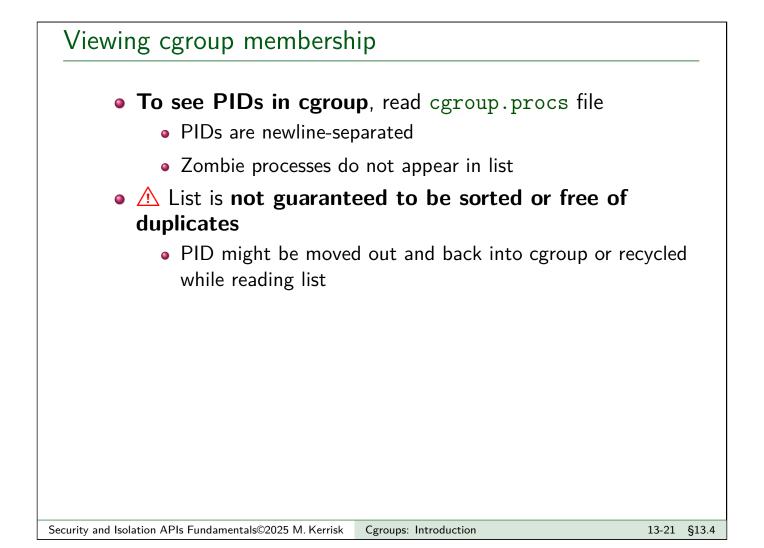
 Relationships between cgroups are reflected by creating nested (arbitrarily deep) subdirectory structure

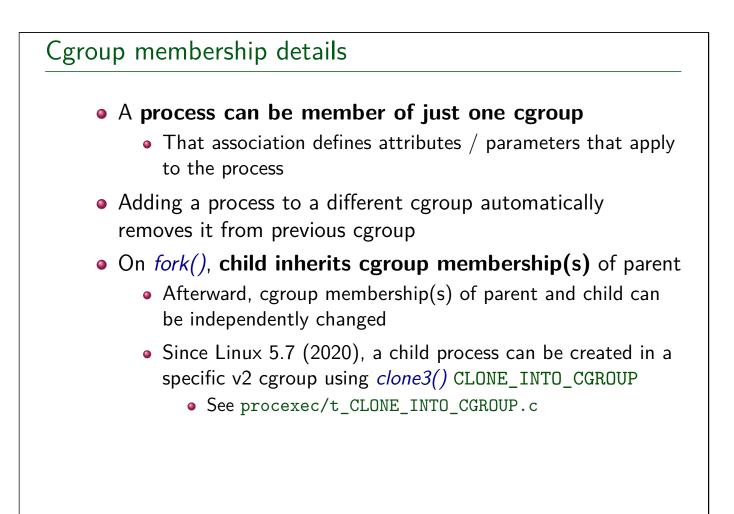


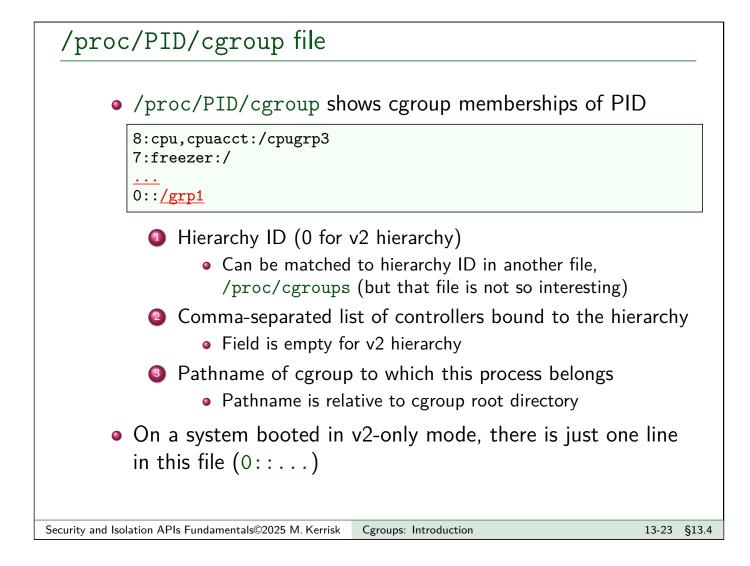
• To move a **process** to a cgroup, we write its PID to cgroup.procs file in corresponding subdirectory

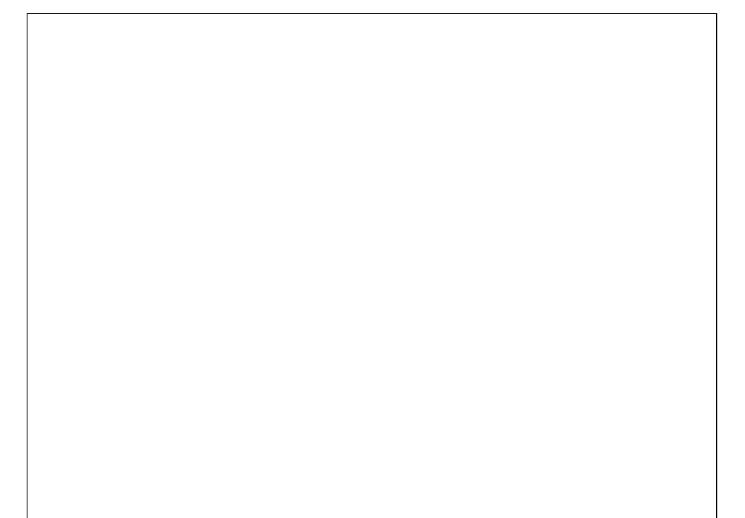
echo \$\$ > /sys/fs/cgroup/mygrp/cgroup.procs

- In multithreaded process, moves all threads to cgroup
- 🛆 Can write only one PID at a time
 - Otherwise, write() fails with EINVAL





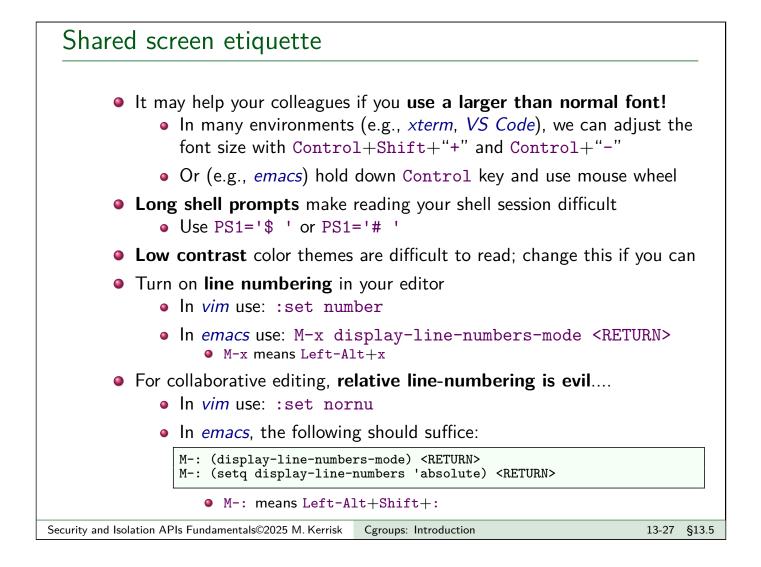


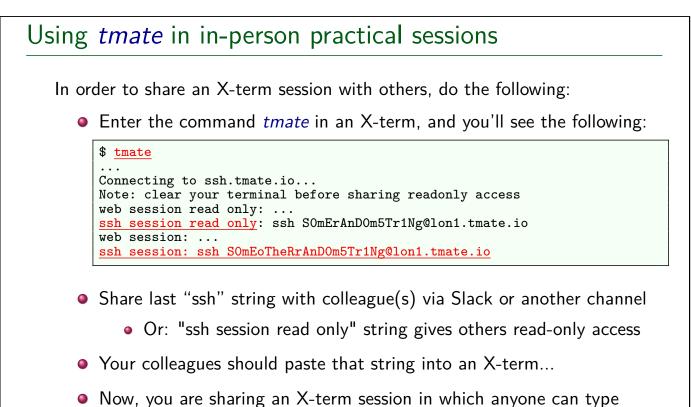


Outline Cgroups: Introduction 13 13-1 13 - 313.1 Preamble 13.2 What are control groups? 13-613.3 An example: the pids controller 13 - 1313.4 Creating, destroying, and populating a cgroup 13-17 13.5 Exercises 13 - 2413.6 Enabling and disabling controllers 13 - 3113.7 Exercises 13 - 46

Notes for online practical sessions

- Small groups in breakout rooms
 - Write a note into Slack if you have a preferred group
- We will go faster, if groups collaborate on solving the exercise(s)
 - You can share a screen in your room
- I will circulate regularly between rooms to answer questions
- Zoom has an "Ask for help" button...
- Keep an eye on the #general Slack channel
 - Perhaps with further info about exercise;
 - Or a note that the exercise merges into a break
- When your room has finished, write a message in the Slack channel: "***** Room X has finished *****"
 - Then I have an idea of how many people have finished





- Any "mate" can cut the connection to the session with the
- To see above message again: tmate show-messages

3-character sequence <ENTER> \sim .

Booting to cgroups v2 • In preparation for the following exercises, if necessary reboot your system to use cgroups v2 only, as follows... • First, check whether your system is already booted to use cgroups v2 only: \$ grep cgroup2 /proc/mounts # Is there a v2 mount? cgroup2 /sys/fs/cgroup cgroup2 ... \$ grep cgroup /proc/mounts | grep -v name= | grep -vc cgroup2 # 0 == no v1 controllers are mounted • If there is a v2 mount, and no v1 controllers are mounted, then you do not need to do anything further; otherwise: • From the GRUB boot menu, you can boot to cgroups v2-only mode by editing the boot command (select a GRUB menu entry and type "e"). In the line that begins with "linux", add the following parameter: systemd.unified_cgroup_hierarchy 13-29 §13.5 Security and Isolation APIs Fundamentals©2025 M. Kerrisk Cgroups: Introduction

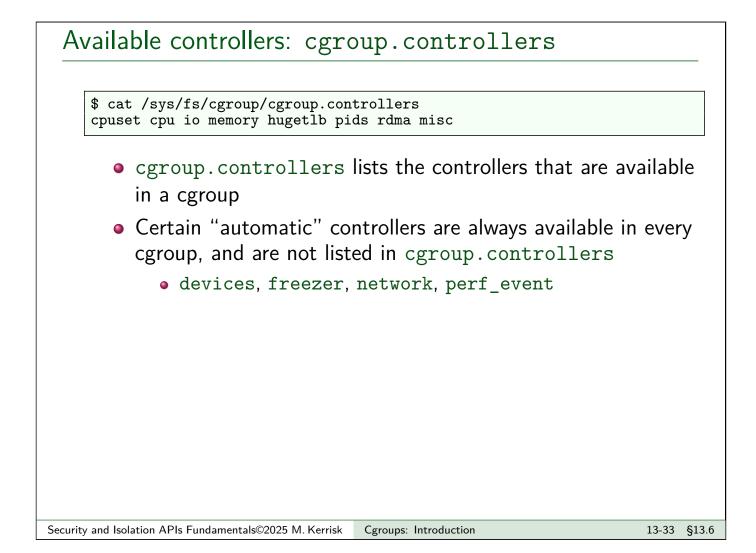
Exercises	
then	is exercise, we create a cgroup, place a process in the cgroup, and migrate that process to a different cgroup. Create two subdirectories, m1 and m2, in the cgroup root directory (/sys/fs/cgroup).
٩	Execute the following command, and note the PID assigned to the resulting process:
	# 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.
٥	Now write the PID of the process into the file m2/cgroup.procs.
٥	Is the PID still visible in the file m1/cgroup.procs? Explain.
۹	Try removing cgroup m1 using the command rm -rf m1. Why doesn't this work?
۹	If it is still running, kill the <i>sleep</i> process and then remove the cgroups m1 and m2 using the <i>rmdir</i> command.

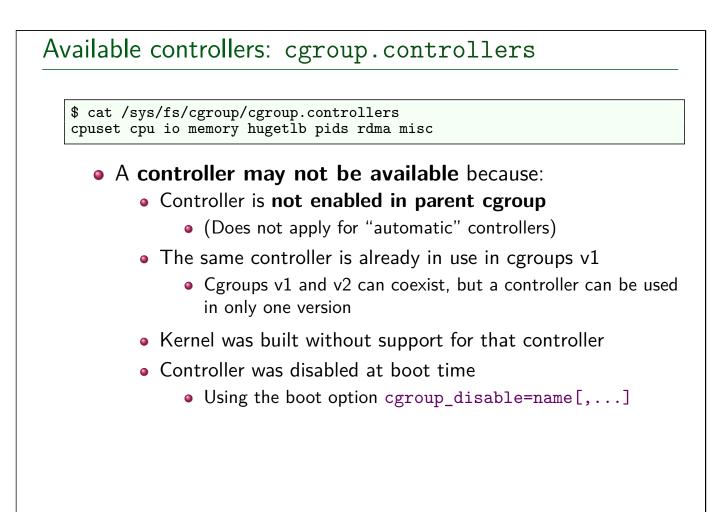
Outline	
13 Cgroups: Introduction	13-1
13.1 Preamble	13-3
13.2 What are control groups?	13-6
13.3 An example: the pids controller	13-13
13.4 Creating, destroying, and populating a cgroup	13-17
13.5 Exercises	13-24
13.6 Enabling and disabling controllers	13-31
13.7 Exercises	13-46

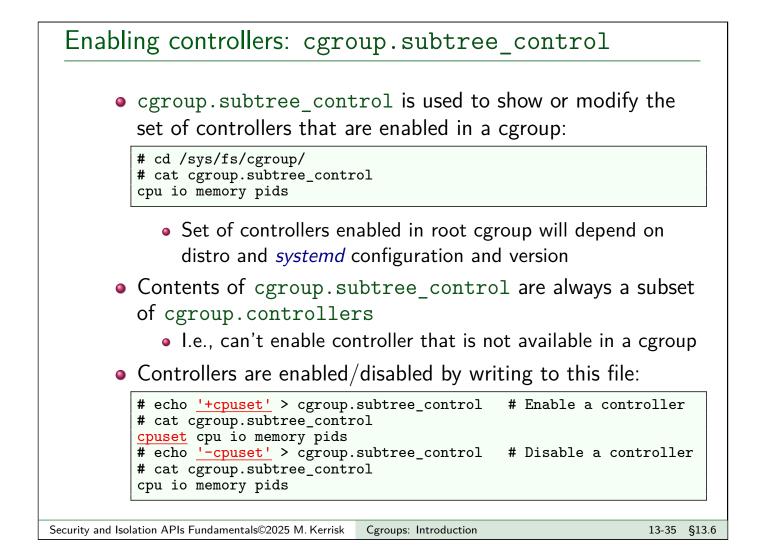
Enabling and disabling controllers

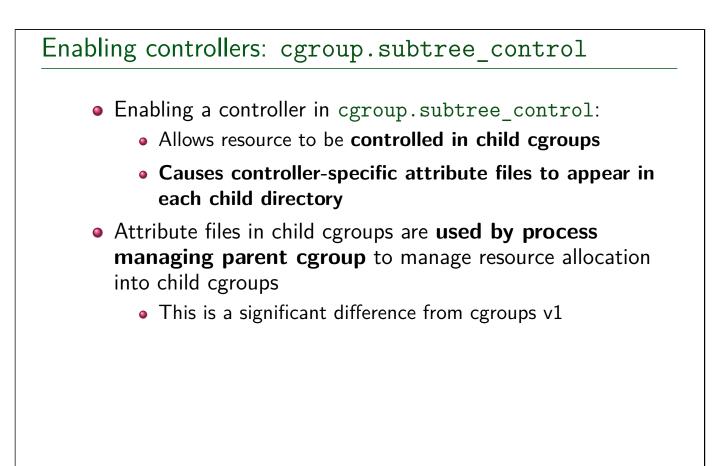
• Each cgroup v2 directory contains two files:

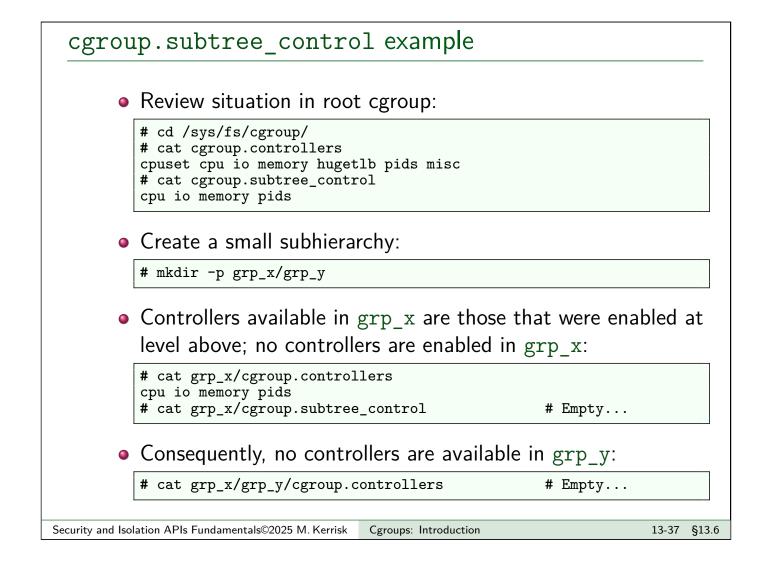
- cgroup.controllers: lists controllers that are **available** in this cgroup
- cgroup.subtree_control: used to list/modify set of controllers that are **enabled** in this cgroup
 - Always a subset of cgroup.controllers
- Together, these files allow different controllers to be managed to **different levels of granularity** in v2 hierarchy

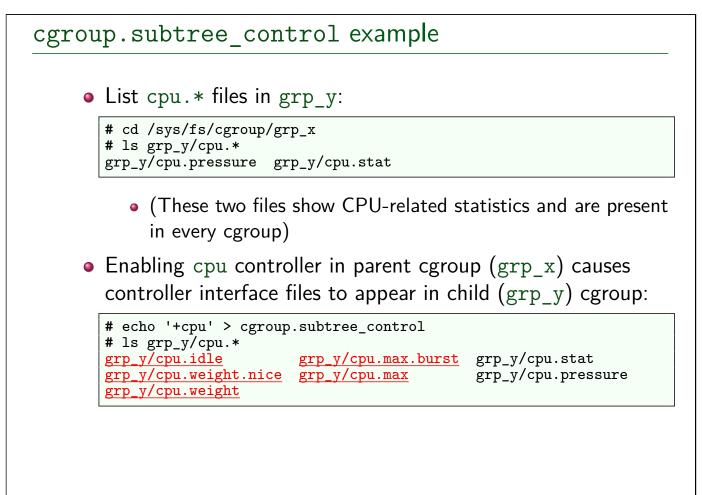


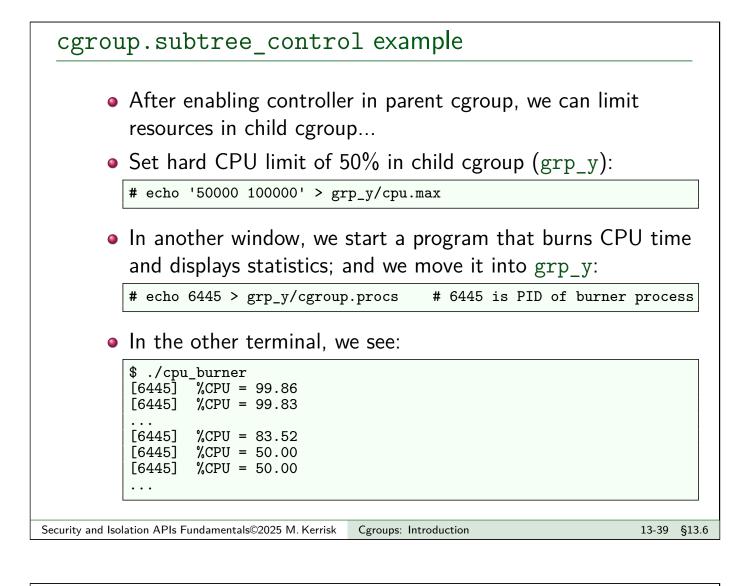


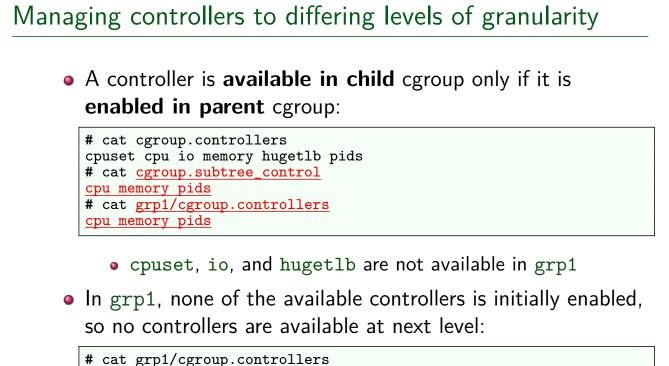










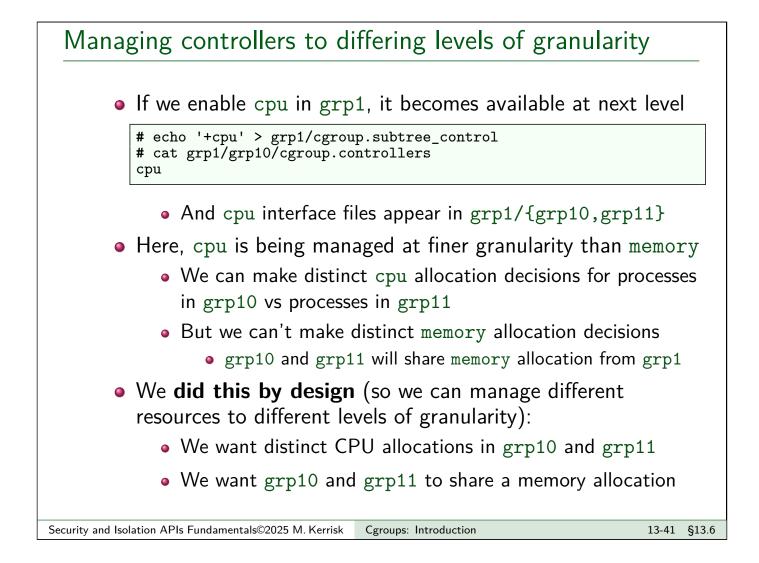


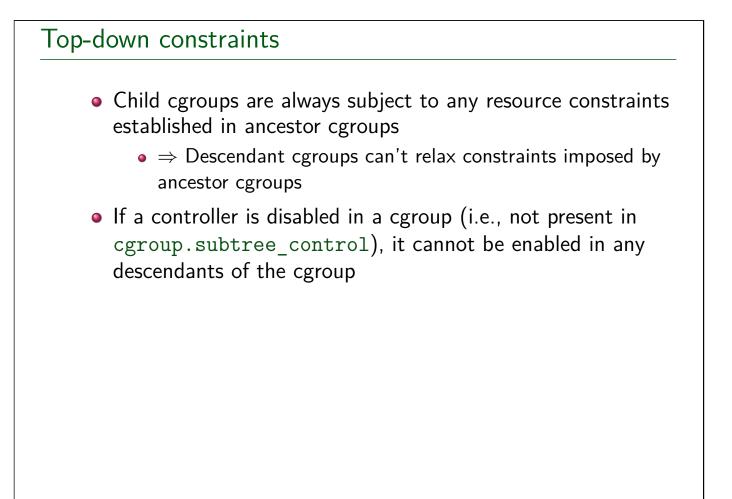
```
cpu memory pids
# cat grp1/cgroup.subtree_control
# mkdir grp1/{grp10,grp11}
# cat grp1/grp2/cgroup.controllers
```

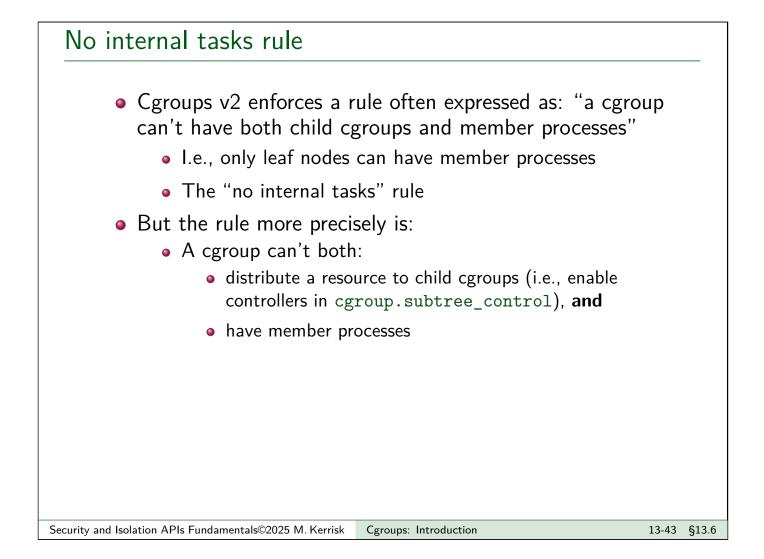
```
# Empty
```

Make grandchild cgroups

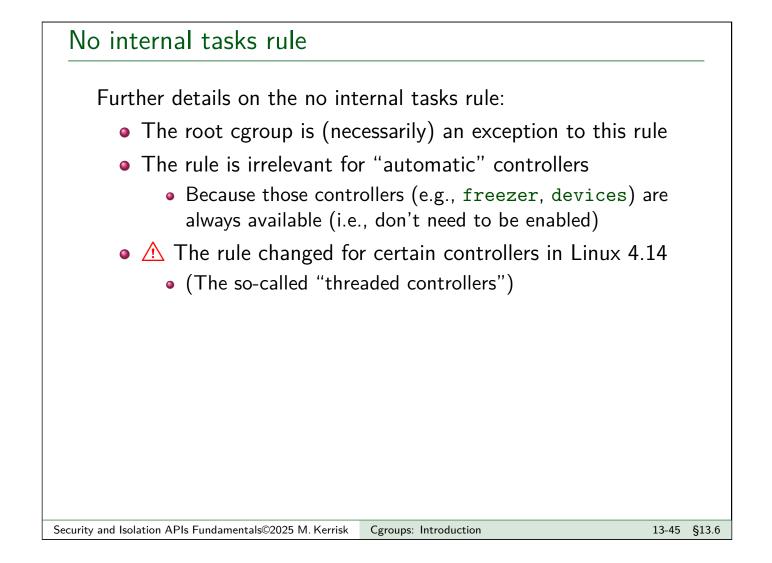
```
# Empty
```







No internal tasks rule Revised statement: "A cgroup can't both distribute resources and have member processes" Conversely (1): A cgroup can have member processes and child cgroups... if it does not enable controllers for child cgroups Conversely (2): If cgroup has child cgroups and processes, the processes must be moved elsewhere before enabling controllers E.g., processes could be moved to child cgroups

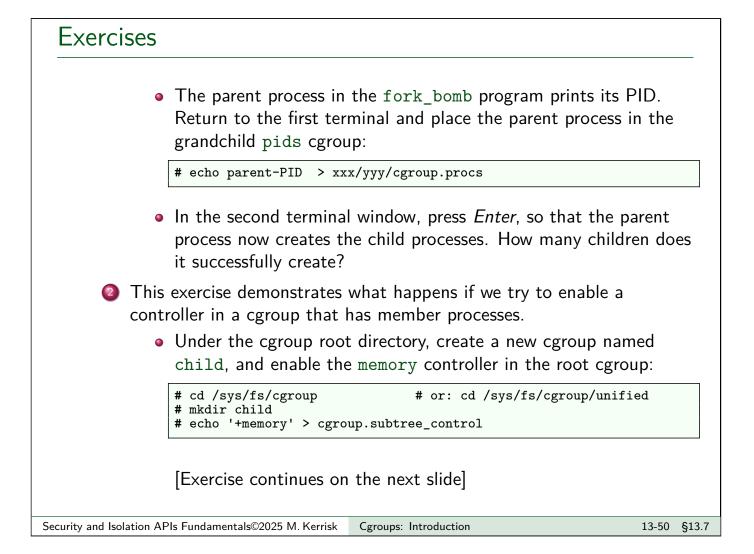


Outline 13 Cgroups: Introduction 13-1 13.1 Preamble 13-3 13.2 What are control groups? 13-613.3 An example: the pids controller 13-13 13.4 Creating, destroying, and populating a cgroup 13-17 13.5 Exercises 13-24 13.6 Enabling and disabling controllers 13-31 13.7 Exercises 13-46

Exercises

	lown fashion, using the cgroups v2 pids controller. To simplify the following steps, change your current directory to the cgroup root directory (i.e., the location where the cgroup2 filesystem is mounted; on recent <i>systemd</i> -based systems, this will be /sys/fs/cgroup, or possibly /sys/fs/cgroup/unified).
۰	Create a child and grandchild directory in the cgroup filesystem and enable the PIDs controller in the root directory and the first subdirectory:
	<pre># mkdir xxx # mkdir xxx/yyy # echo '+pids' > cgroup.subtree_control # echo '+pids' > xxx/cgroup.subtree_control</pre>
	[Exercise continues on next page]

Exercises • Set an upper limit of 10 tasks in the child cgroup, and an upper limit of 20 tasks in the grandchild cgroup: # echo '10' > xxx/pids.max # echo '20' > xxx/yyy/pids.max • In another terminal, use the supplied cgroups/fork bomb.c program. fork_bomb <num-children> [<child-sleep>] # Default: 0 300 Run the program with the following command line, which (after the user presses *Enter*) will cause the program to create 30 children that sleep for (the default) 300 seconds: \$./fork_bomb 30 [Exercise continues on next page...] 13-49 §13.7 Security and Isolation APIs Fundamentals©2025 M. Kerrisk Cgroups: Introduction



Exercises

• Start a process running *sleep*, and place the process into the child cgroup:

sleep 1000 &
echo \$! > child/cgroup.procs

• What happens if we now try to enable the memory controller in the child cgroup via the following command?

```
# echo '+memory' > child/cgroup.subtree_control
```

• Does the result differ if we reverse the order of the preceding steps (i.e., enable the controller, then place a process in the cgroup)?

Security and Isolation APIs Fundamentals©2025 M. Kerrisk

Cgroups: Introduction

13-51 §13.7

lotes			