Linux Security and Isolation APIs

Control Groups (cgroups): Introduction

Michael Kerrisk, man7.org © 2025

January 2025

mtk@man7.org

Outline	Rev: #8d7fc39ab521
18 Cgroups: Introduction	18-1
18.1 Preamble	18-3
18.2 What are control groups?	18-9
18.3 An example: the pids controller	18-16
18.4 Creating, destroying, and populating a cgroup	18-21
18.5 Exercises	18-29
18.6 Enabling and disabling controllers	18-35
18.7 Exercises	18-50

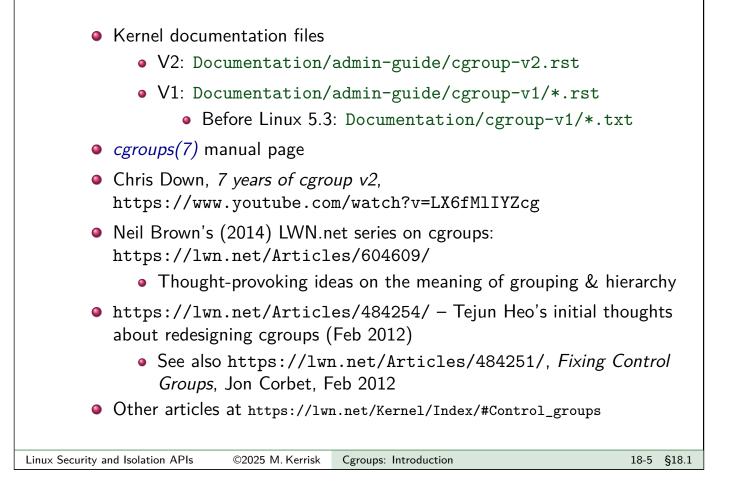
Outline Cgroups: Introduction 18 18-1 18.1 Preamble 18 - 318.2 What are control groups? 18 - 918.3 An example: the pids controller 18 - 1618.4 Creating, destroying, and populating a cgroup 18-21 18.5 Exercises 18 - 2918.6 Enabling and disabling controllers 18 - 3518.7 Exercises 18 - 50

Goals

• We'll focus on:

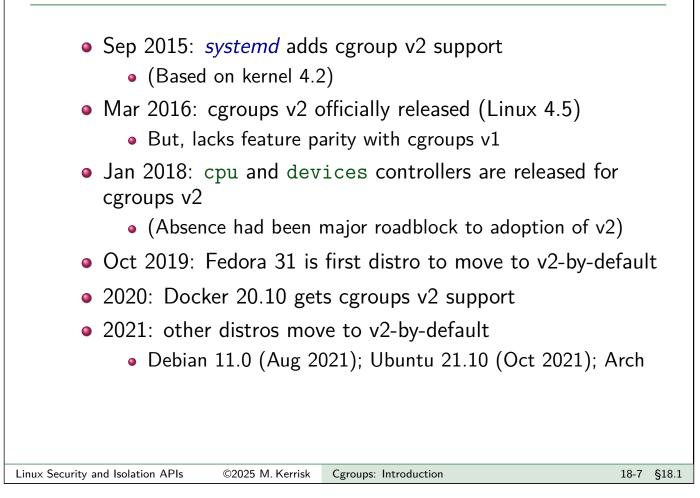
- General principles of operation; goals of cgroups
- The cgroup2 filesystem
- Interacting with cgroup2 filesystem using shell commands
- Origin of cgroups v2 (i.e., problems with cgroups v1)
- Differences between cgroups v2 and v1
- We'll look briefly at some of the controllers

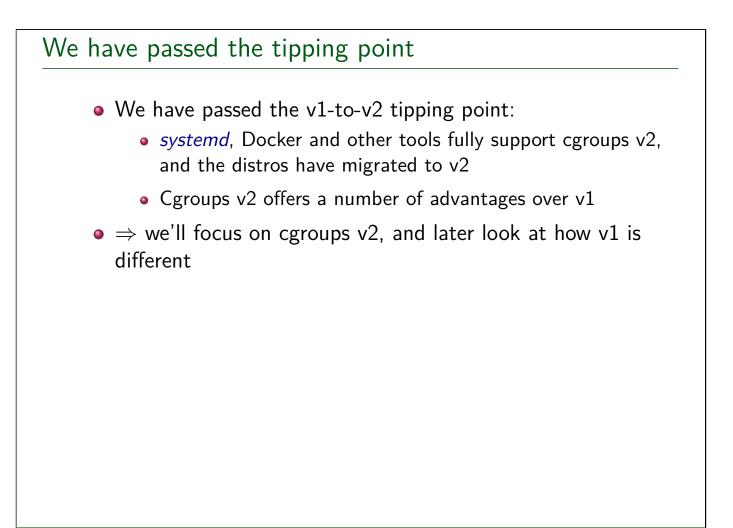
Resources



Some history 2006/2007, "Process Containers" @ Google ⇒ Cgroups v1 Jan 2008: initial mainline kernel release (Linux 2.6.24) Three resource controllers (all CPU-related) in initial release Subsequently, other controllers are added memory, devices, freezer, net_cls, blkio... But a few years of uncoordinated design leads to a mess Decentralized design fails us... again 2012: work has already begun on cgroups v2...

Some history



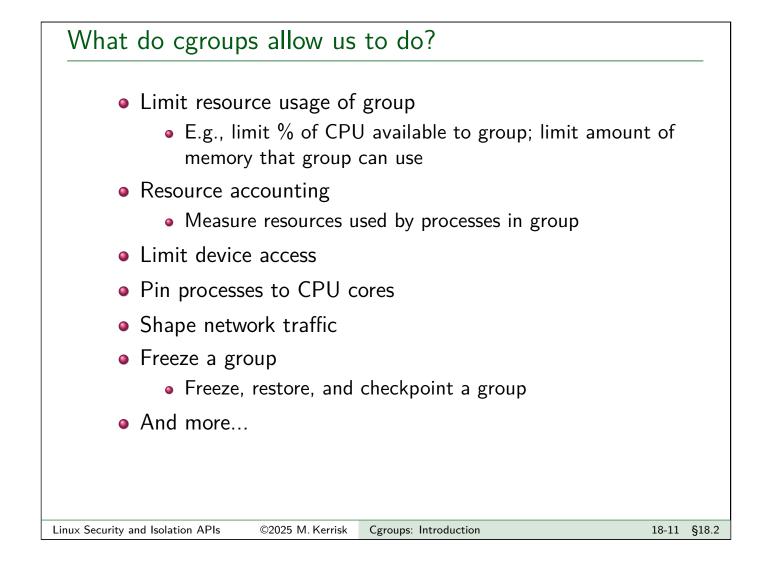


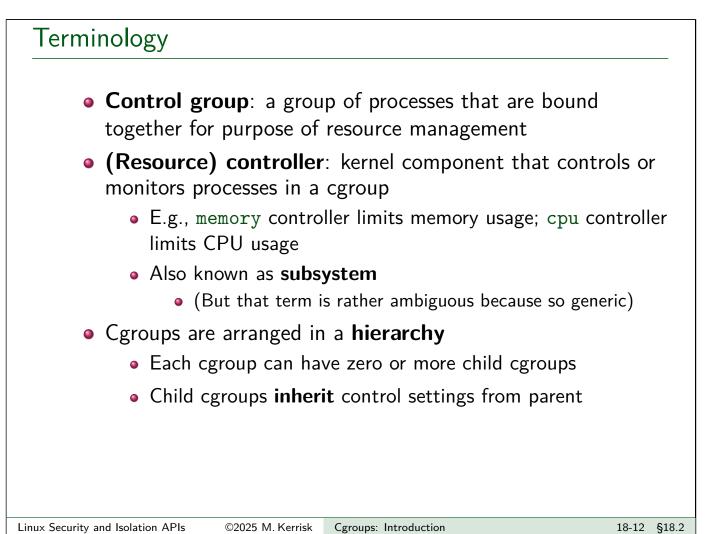
Outline

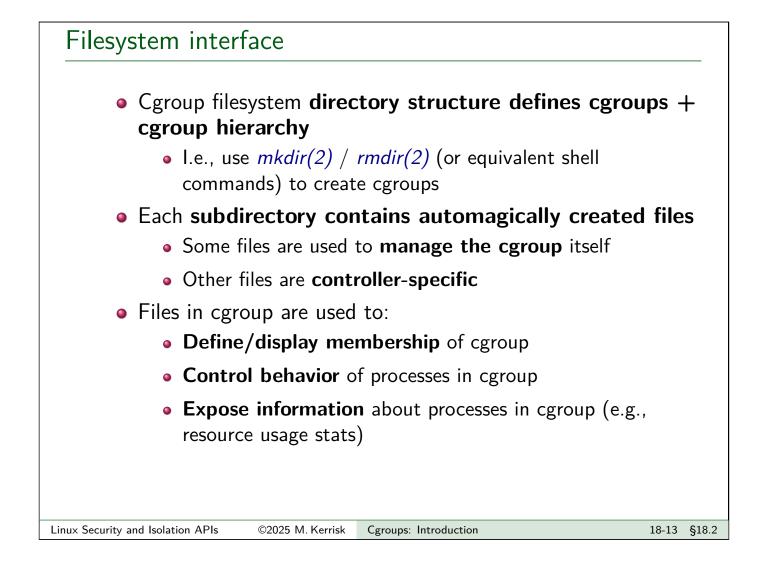
18 Cgroups: Introduction	18-1
18.1 Preamble	18-3
18.2 What are control groups?	18-9
18.3 An example: the pids controller	18-16
18.4 Creating, destroying, and populating a cgroup	18-21
18.5 Exercises	18-29
18.6 Enabling and disabling controllers	18-35
18.7 Exercises	18-50

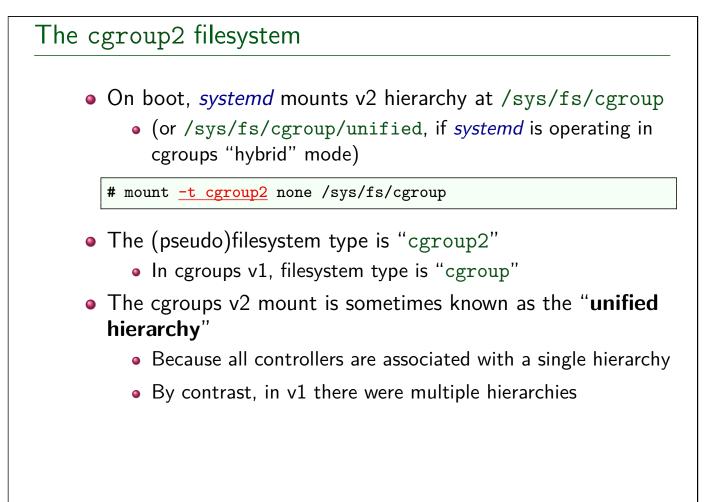
What are control groups?

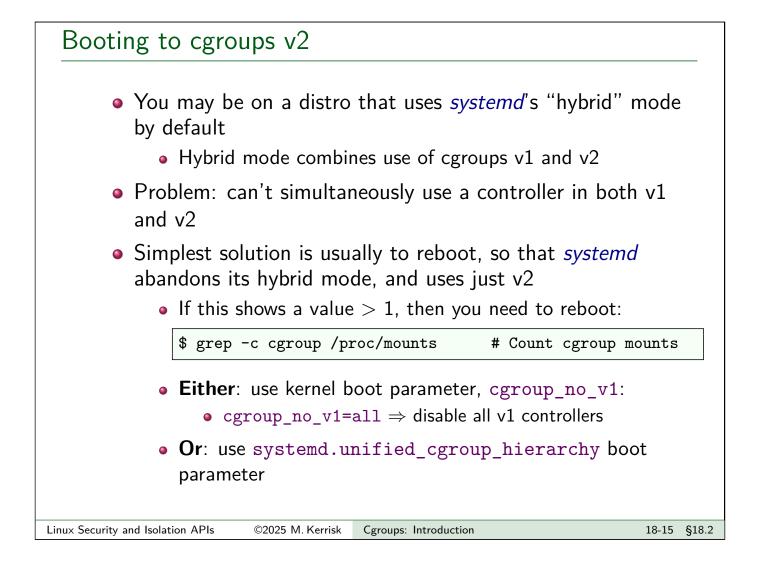
- Two principal components:
 - A mechanism for hierarchically grouping processes
 - A set of **controllers** (kernel components) that manage, control, or monitor processes in cgroups
- Interface is via a pseudo-filesystem
- Cgroup manipulation takes form of filesystem operations, which might be done: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











Outline

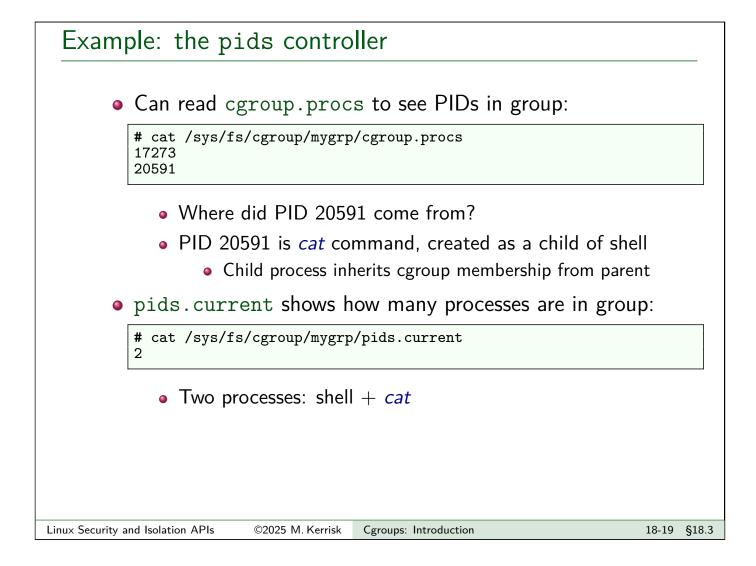
18 Cgroups: Introduction	18-1
18.1 Preamble	18-3
18.2 What are control groups?	18-9
18.3 An example: the pids controller	18-16
18.4 Creating, destroying, and populating a cgroup	18-21
18.5 Exercises	18-29
18.6 Enabling and disabling controllers	18-35
18.7 Exercises	18-50

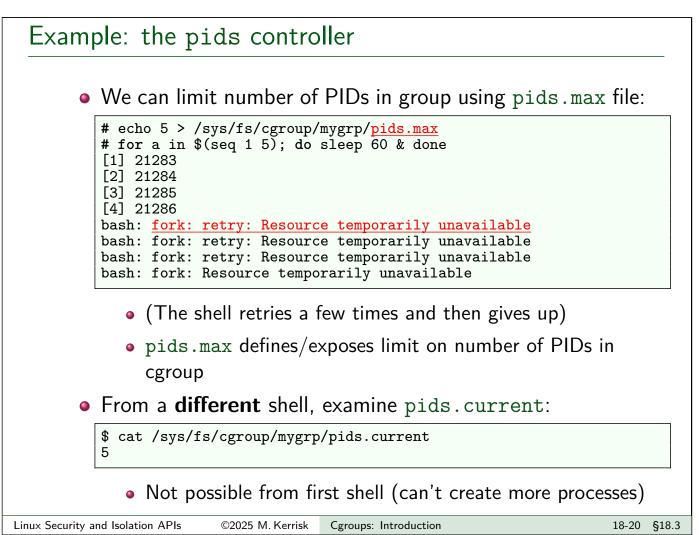
Example: the pids controller

- 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





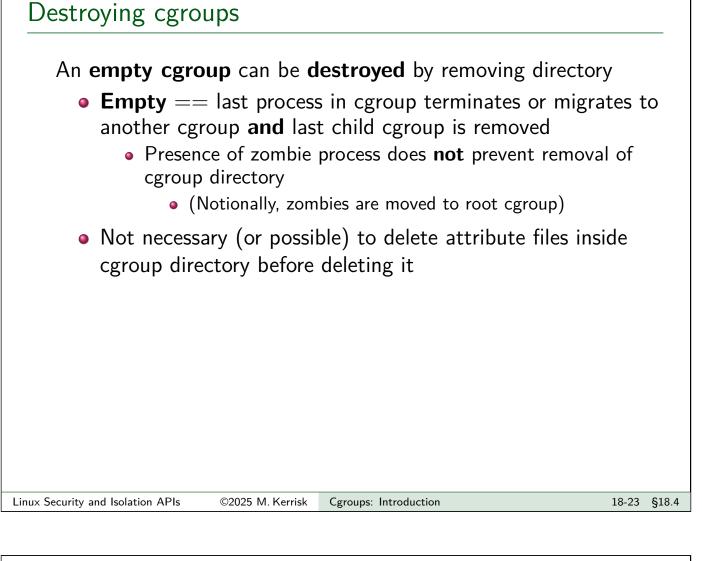
Outline Cgroups: Introduction 18 18-1 18.1 Preamble 18 - 318.2 What are control groups? 18 - 918.3 An example: the pids controller 18 - 1618.4 Creating, destroying, and populating a cgroup 18-21 18.5 Exercises 18 - 2918.6 Enabling and disabling controllers 18 - 3518.7 Exercises 18 - 50

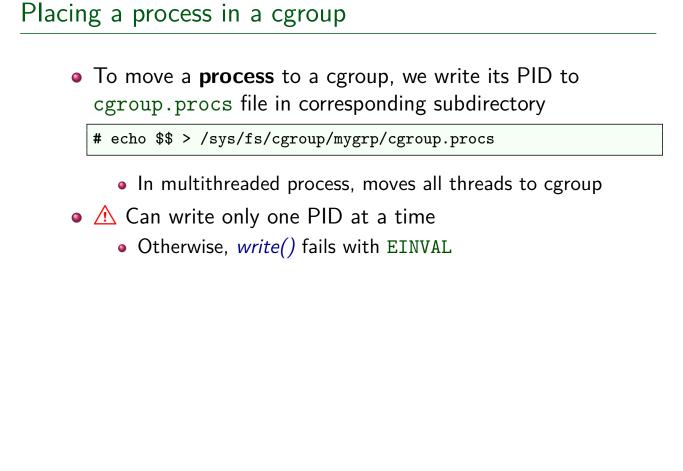
Creating cgroups

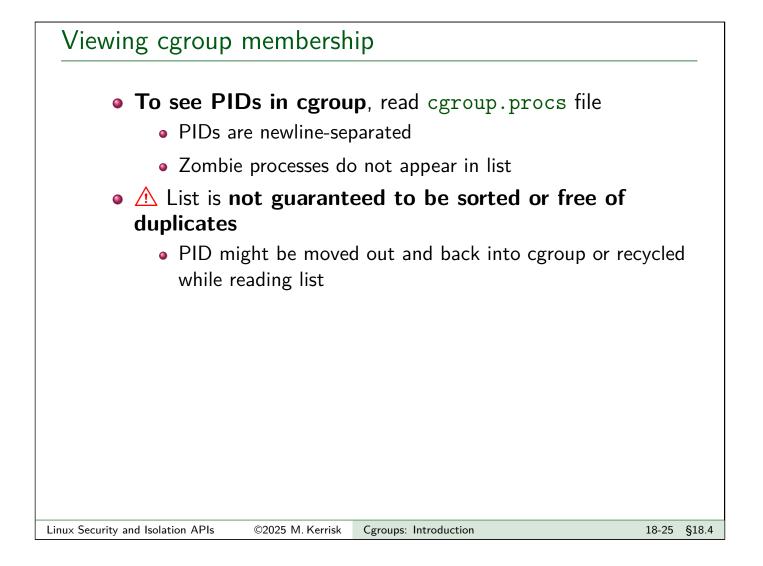
- Initially, all processes on system are members of root cgroup
- New cgroups are **created** by creating subdirectories under cgroup mount point:

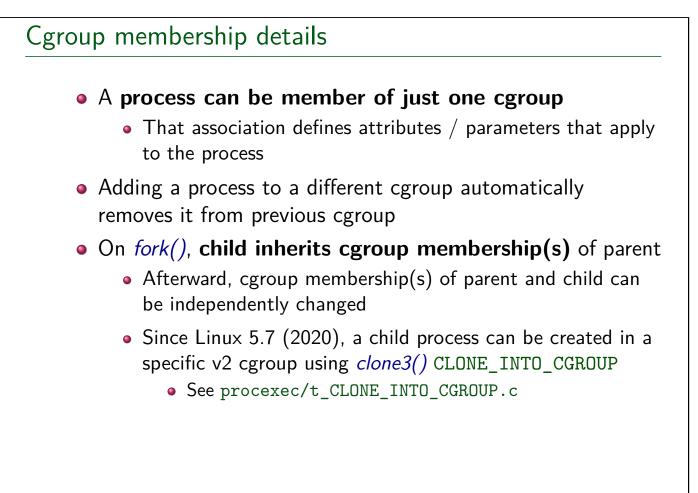
```
# mkdir /sys/fs/cgroup/mygrp
```

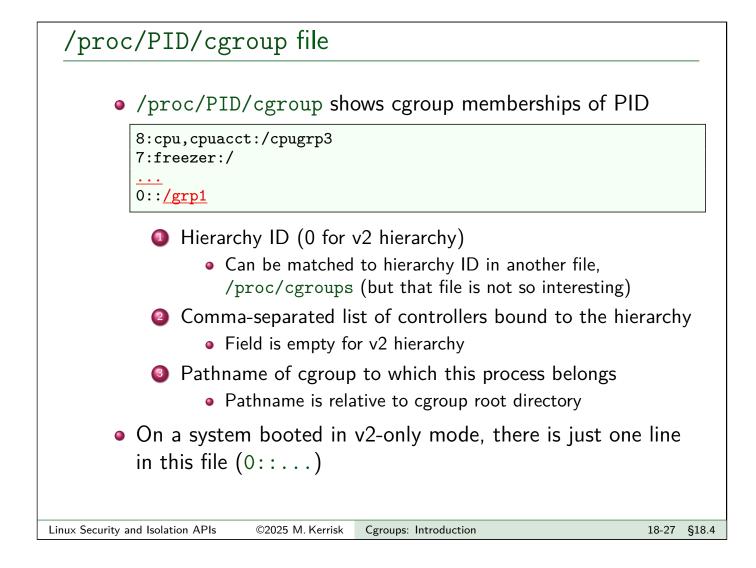
 Relationships between cgroups are reflected by creating nested (arbitrarily deep) subdirectory structure

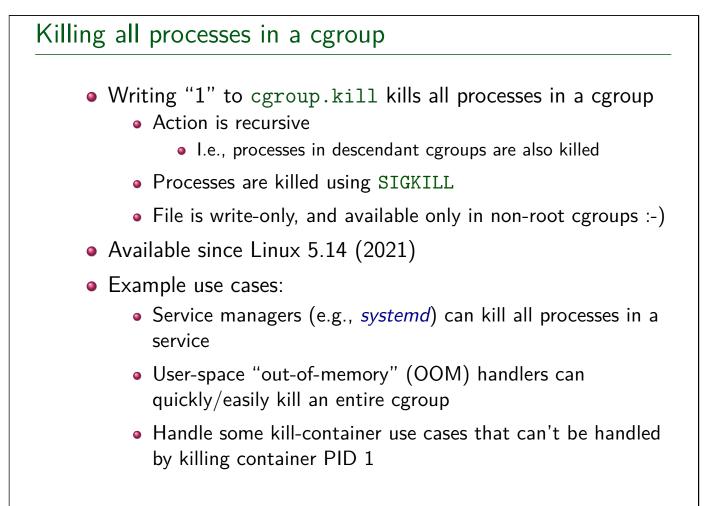








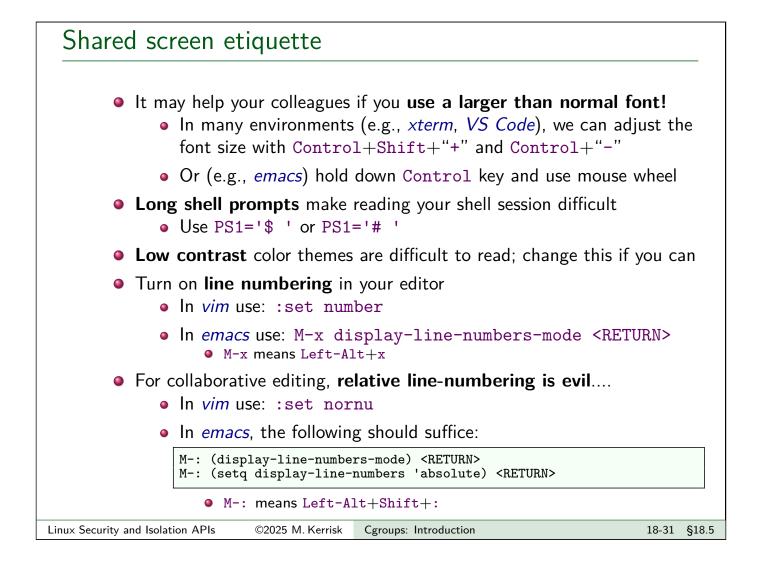


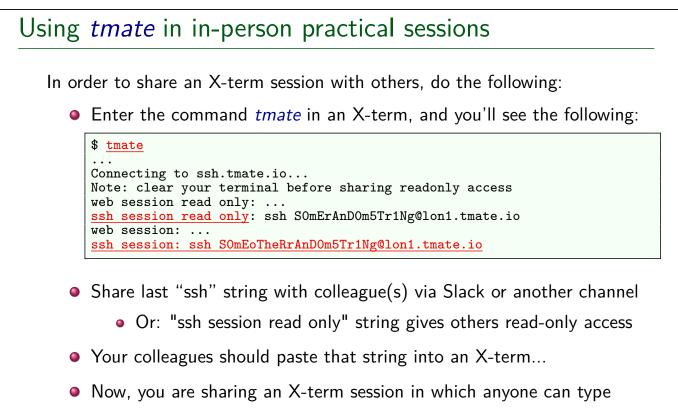


Outline Cgroups: Introduction 18 18-1 18.1 Preamble 18 - 318.2 What are control groups? 18 - 918.3 An example: the pids controller 18 - 1618.4 Creating, destroying, and populating a cgroup 18-21 18.5 Exercises 18 - 2918.6 Enabling and disabling controllers 18 - 3518.7 Exercises 18 - 50

Notes for online practical sessions

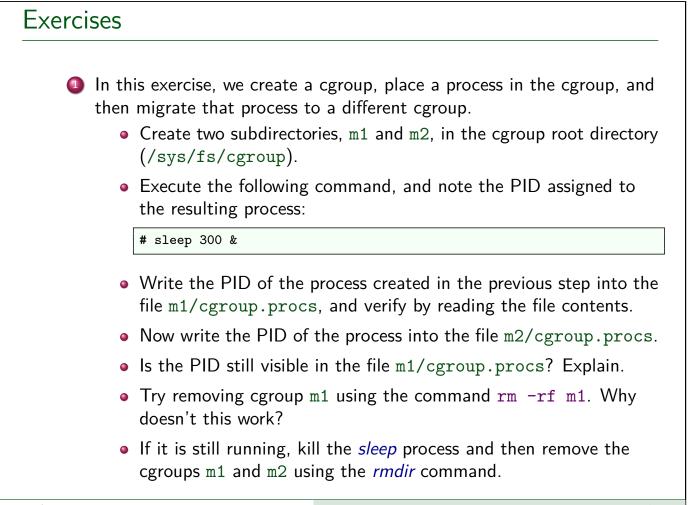
- Small groups in breakout rooms
 - Write a note into Slack if you have a preferred group
- We will go faster, if groups collaborate on solving the exercise(s)
 - You can share a screen in your room
- I will circulate regularly between rooms to answer questions
- Zoom has an "Ask for help" button...
- 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 3-character sequence <ENTER> \sim .
- To see above message again: tmate show-messages

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 18-33 §18.5 Linux Security and Isolation APIs ©2025 M. Kerrisk Cgroups: Introduction

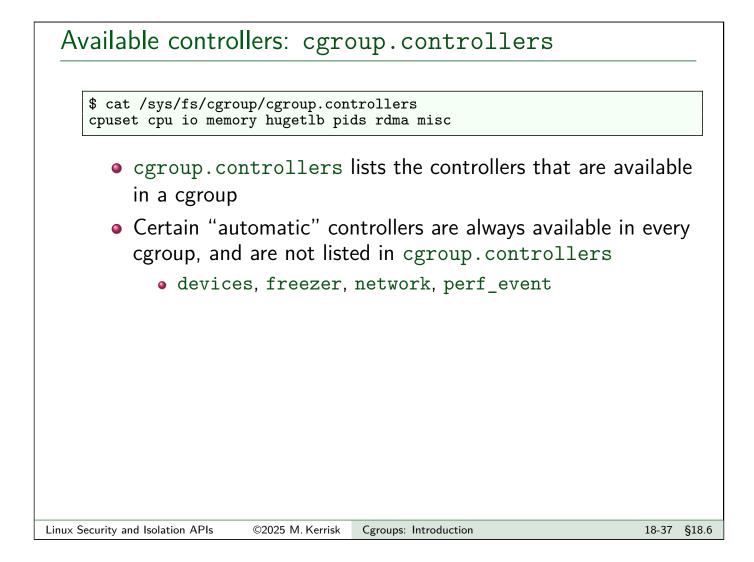


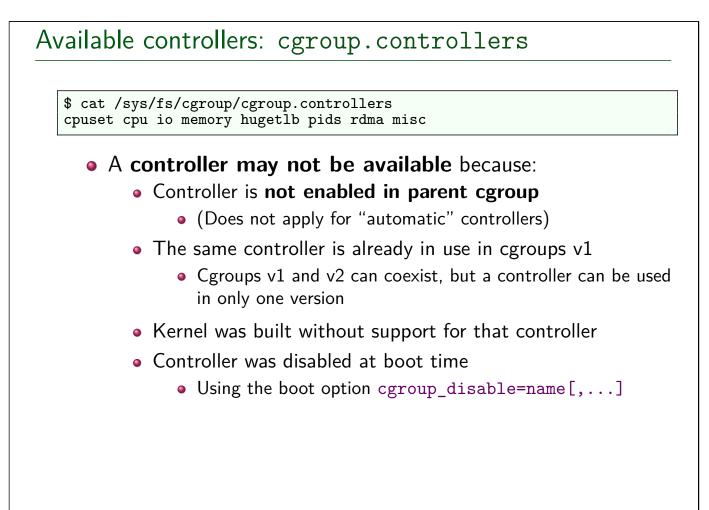
Outline	
18 Cgroups: Introduction	18-1
18.1 Preamble	18-3
18.2 What are control groups?	18-9
18.3 An example: the pids controller	18-16
18.4 Creating, destroying, and populating a cgroup	18-21
18.5 Exercises	18-29
18.6 Enabling and disabling controllers	18-35
18.7 Exercises	18-50

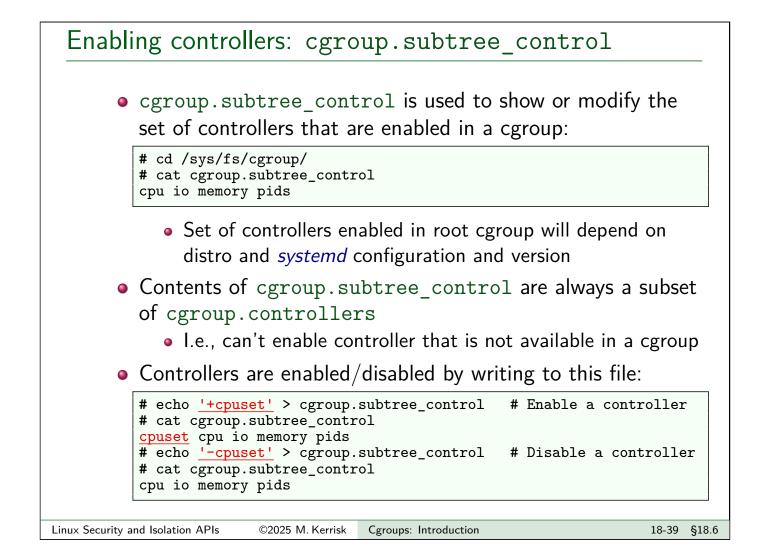
Enabling and disabling controllers

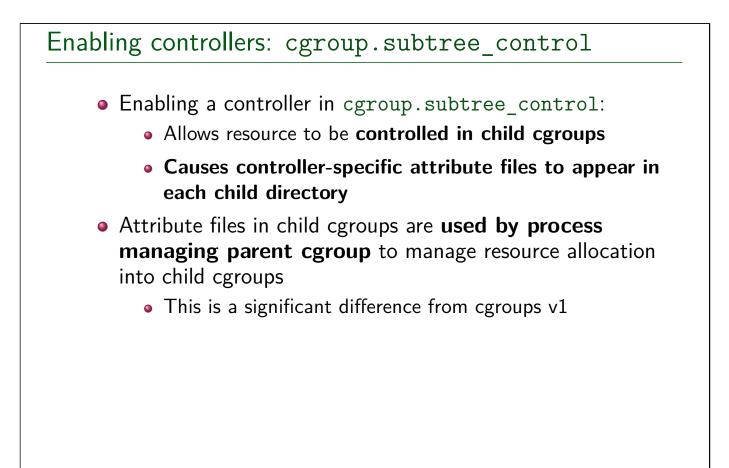
• Each cgroup v2 directory contains two files:

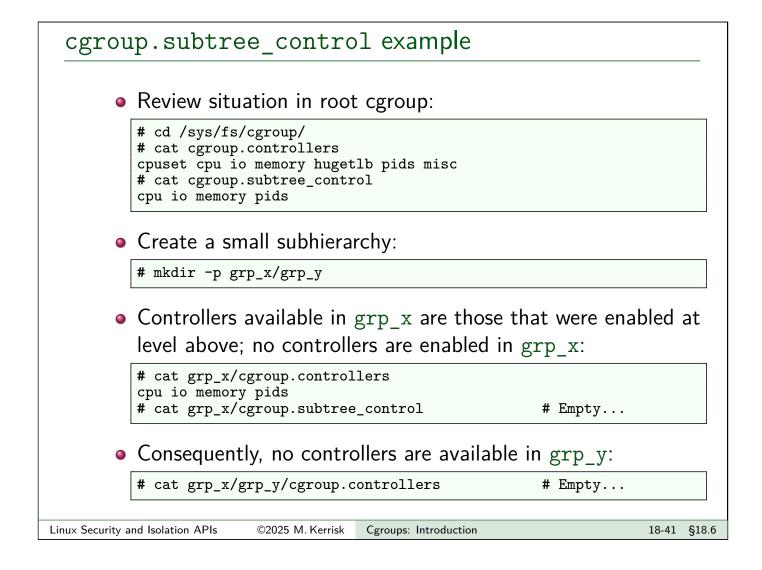
- cgroup.controllers: lists controllers that are **available** in this cgroup
- cgroup.subtree_control: used to list/modify set of controllers that are **enabled** in this cgroup
 - Always a subset of cgroup.controllers
- Together, these files allow different controllers to be managed to **different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

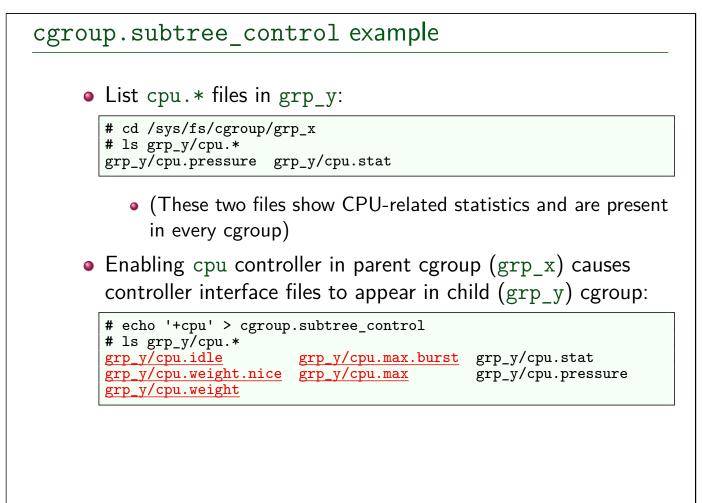


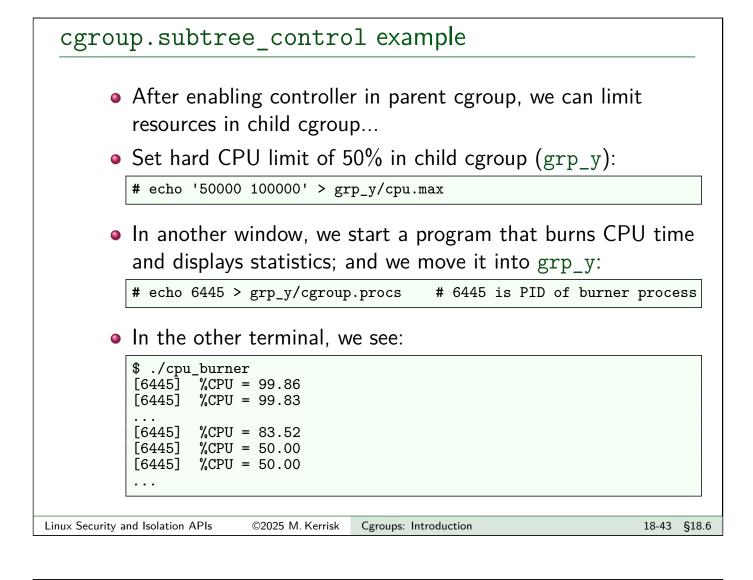


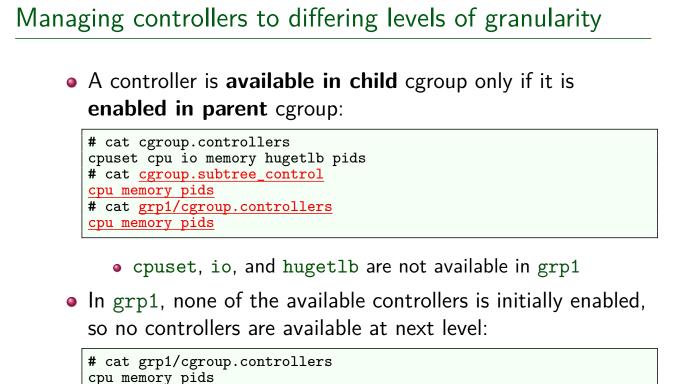










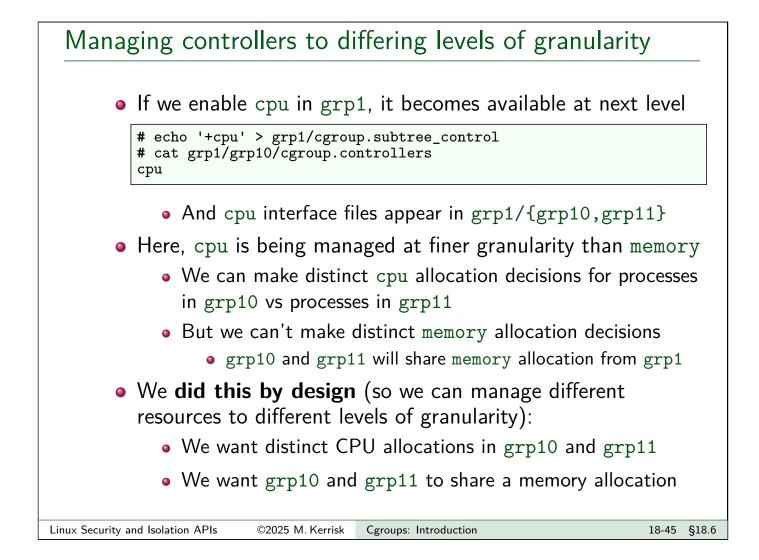


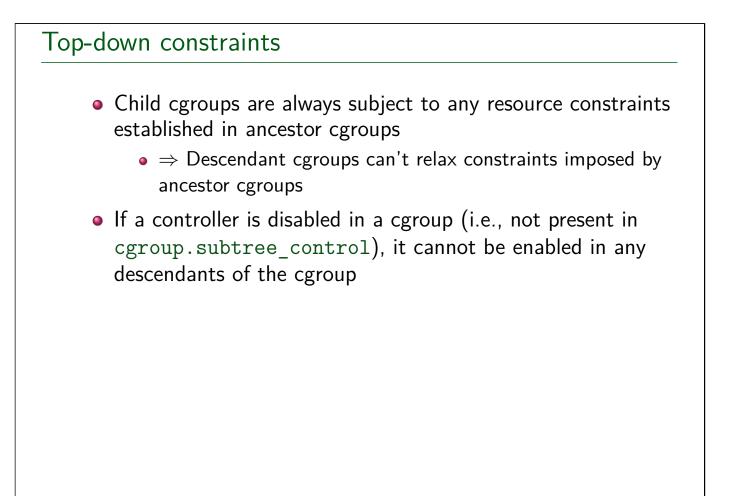
```
# cat grp1/cgroup.subtree_control
# mkdir grp1/{grp10,grp11}
# cat grp1/grp2/cgroup.controllers
```

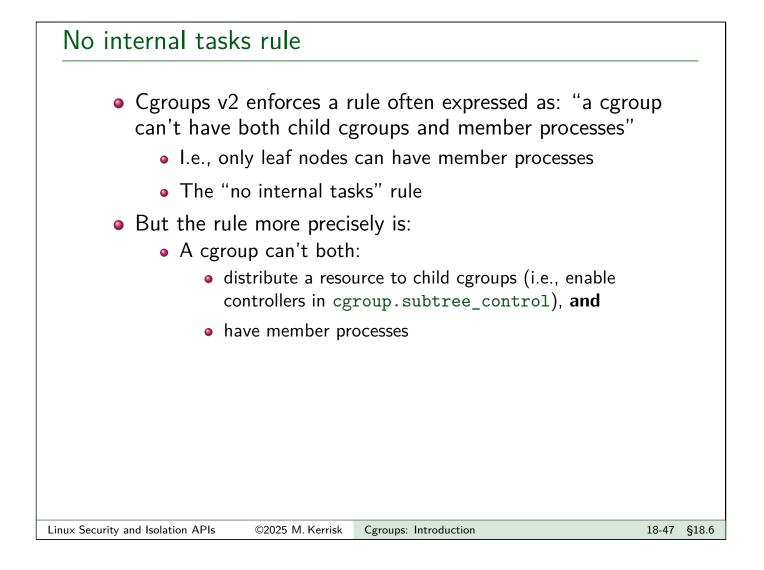
Empty
Make grandc

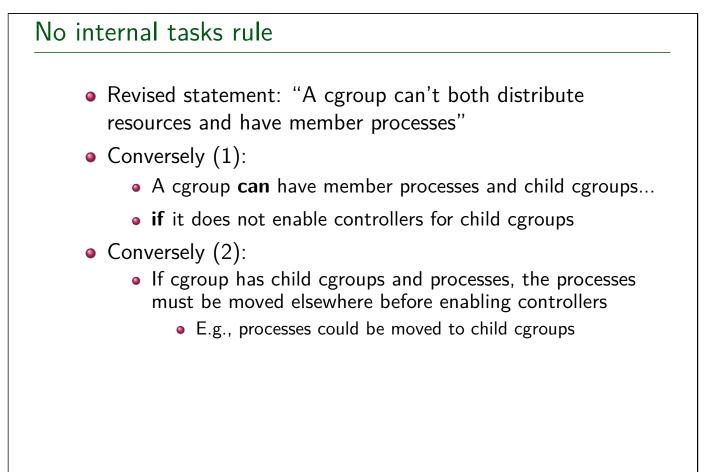
Make grandchild cgroups
Empty

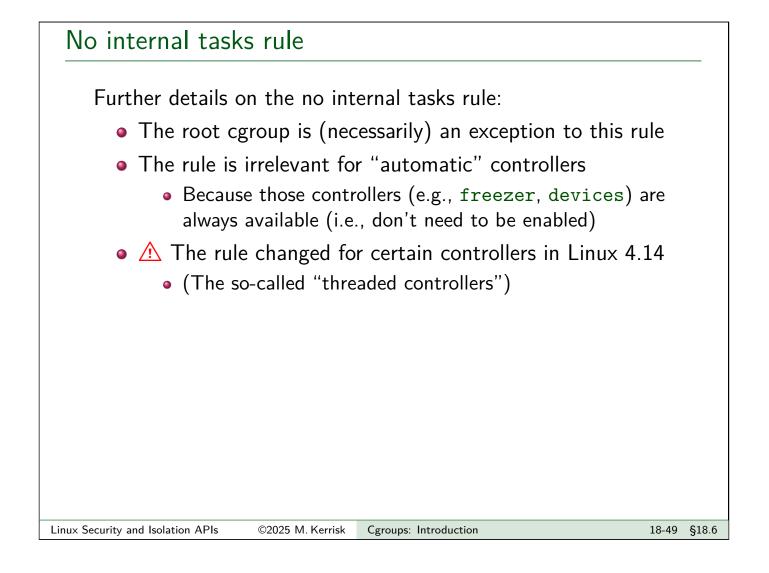
```
# Empty
```











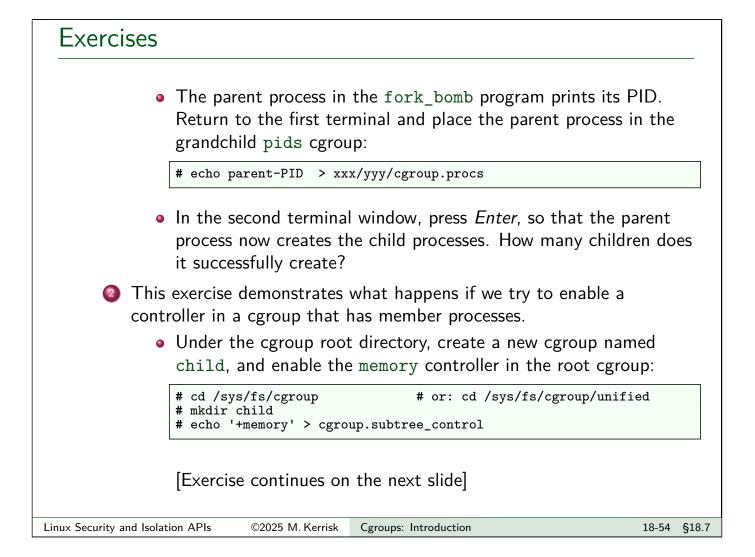
Outline 18 Cgroups: Introduction

18	Cgroups: Introduction	18-1
18.1	Preamble	18-3
18.2	What are control groups?	18-9
18.3	An example: the pids controller	18-16
18.4	Creating, destroying, and populating a cgroup	18-21
18.5	Exercises	18-29
18.6	Enabling and disabling controllers	18-35
18.7	Exercises	18-50

Exercises

-	exercise demonstrates that resource constraints apply in a down fashion, using the cgroups v2 pids controller.
٩	Check that the pids controller is visible in the cgroup root cgroup.controllers file. If it is not, reboot the kernel as described on slide 18-15.
•	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).
٥	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...] Linux Security and Isolation APIs 18-53 §18.7 ©2025 M. Kerrisk Cgroups: Introduction



Exercises Start a process running *sleep*, and place the process into the child cgroup: # sleep 1000 & # 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)?

Linux Security and Isolation APIs

©2025 M. Kerrisk Cgr

Cgroups: Introduction

18-55 §18.7

Notes			