

CS438

Namespace

Lecture 12

1st Semester 2021

Kasidit Chanchio

Department of Computer Science

การสร้างภาพเสมือนของระบบลินุกซ์

- ระบบลินุกซ์ประกอบไปด้วยระบบย่อยหลายระบบ
 - ระบบไฟล์
 - ระบบซื้อเครื่อง
 - ระบบเครือข่ายของเครื่อง
 - ระบบจัดการโปรเซส
 - ระบบจัดการการใช้งานทรัพยากร
 - ระบบจัดการผู้ใช้
 - อื่นๆ เช่นระบบจัดการเวลา
- การประมวลผลบนลินุกซ์จัดอยู่ในนามของโปรเซส

การสร้างภาพเสมือนของระบบลินุกซ์

- ระบบลินุกซ์สามารถสร้างภาพเสมือนของส่วนประกอบย่อย ด้วยเนมสเปซ (Namespace)
 - เนมสเปซมีได้หลายอินสแตนซ์ อินสแตนซ์หนึ่งเปรียบเหมือนภาพเสมือนหนึ่ง
- โพรเซสในระบบจะต้องเลือกว่ามันจะรันอยู่ใน เนมสเปซอินสแตนซ์ไหนในส่วนประกอบย่อยอันใดอันหนึ่ง
- เนมสเปซ เป็นการจับกลุ่มของโพรเซส ในเรื่องใดเรื่องหนึ่ง
 - แทนที่จะเป็นกลุ่มเดียว ก็แบ่งแยกออกเป็นหลายกลุ่ม
 - แต่ละกลุ่มมีโลกส่วนตัวของตนเองและโพรเซสคิดว่ากลุ่มของตนเองเป็นผู้ใช้ระบบคอมพิวเตอร์ทั้งหมดไม่สามารถมองเห็นกลุ่มอื่นได้

เปรียบเทียบกับการลงทะเบียน

- เปรียบเหมือนการเรียนรายวิชาของ นศ --> โพรเซส
- แต่ละรายวิชาเป็นเรื่องย่อยที่อยู่ภายใต้หลักสูตร --> ระบบย่อยของไอเอส
- แต่ละ Section เปรียบเหมือน เนมสเปซอินสแตนซ์
- แต่ละวิชามีหลาย sections ที่เป็นภาพเสมือนหลายภาพของวิชาเดียวกัน
- นศ ลงเรียนได้หนึ่ง section เท่านั้น และเห็นเพื่อนใน sec เดียวกันเท่านั้น

รายวิชา	section	section	section	section
TU153	Sec=1, นาย ก นส ข	Sec=2	Sec=3	Sec=5
TU156	Sec=x1 นส ข	Sec=x2	Sec=x3 นาย ก	Sec=x4
CS123	Sec=v1	Sec=v2 นาย ก	Sec=v12 นส ข	Sec=v5
CS337	Sec=a นาย ก	Sec=b	Sec=x	Sec=y นส ข

เนมสเปซ

- เมื่อระบบลินุกซ์รัน จะมี เนมสเปซ เจ็ดชนิด และลินุกซ์จะสร้างอินสแตนซ์ของเนมสเปซทั้งเจ็ดขึ้น เรียกว่า อินสแตนซ์เริ่มต้น (initial instance)
 - ดังนั้นจะมีเนมสเปซอินสแตนซ์ เจ็ดอินสแตนซ์ แต่ละอันมีชนิดที่ต่างกัน
- สำหรับเนมสเปซแต่ละชนิด โพรเซสจะอยู่ในอินสแตนซ์เดียวเท่านั้น
- โพรเซสที่อยู่ในเนมสเปซอินสแตนซ์ จะนึกว่าตนเองเป็นเจ้าของทรัพยากรที่เกี่ยวข้องกับชนิดของเนมสเปซนั้นทั้งหมด
- โพรเซสที่อยู่ในเนมสเปซอินสแตนซ์เดียวกันจะเห็นกัน และเห็นการเปลี่ยนแปลงทรัพยากรที่โพรเซสอื่นในเนมสเปซอินสแตนซ์เดียวกันทำ
- เมื่อ fork โพรเซสลูกจะอยู่ในเนมสเปซอินสแตนซ์เซตเดียวกันกับโพรเซสพ่อแม่
- โพรเซสจะไม่รู้ว่ามีเนมสเปซอินสแตนซ์อื่นรันอยู่ในระบบ

ชนิดของเนมสเปซ

- เม้าท์เนมสเปซ (mount namespace)
- ยูทีเอสเนมสเปซ (UTS namespace)
- ไอพีซีเนมสเปซ (IPC namespace)
- พีไอดีเนมสเปซ (PID namespace)
- เน็ตเวิร์คเนมสเปซ (Network namespace)
- ยูเซอร์เนมสเปซ (User namespace)
- ซีกรุป (Cgroup)
- ไทม์ (Time)

เนมสเปซ

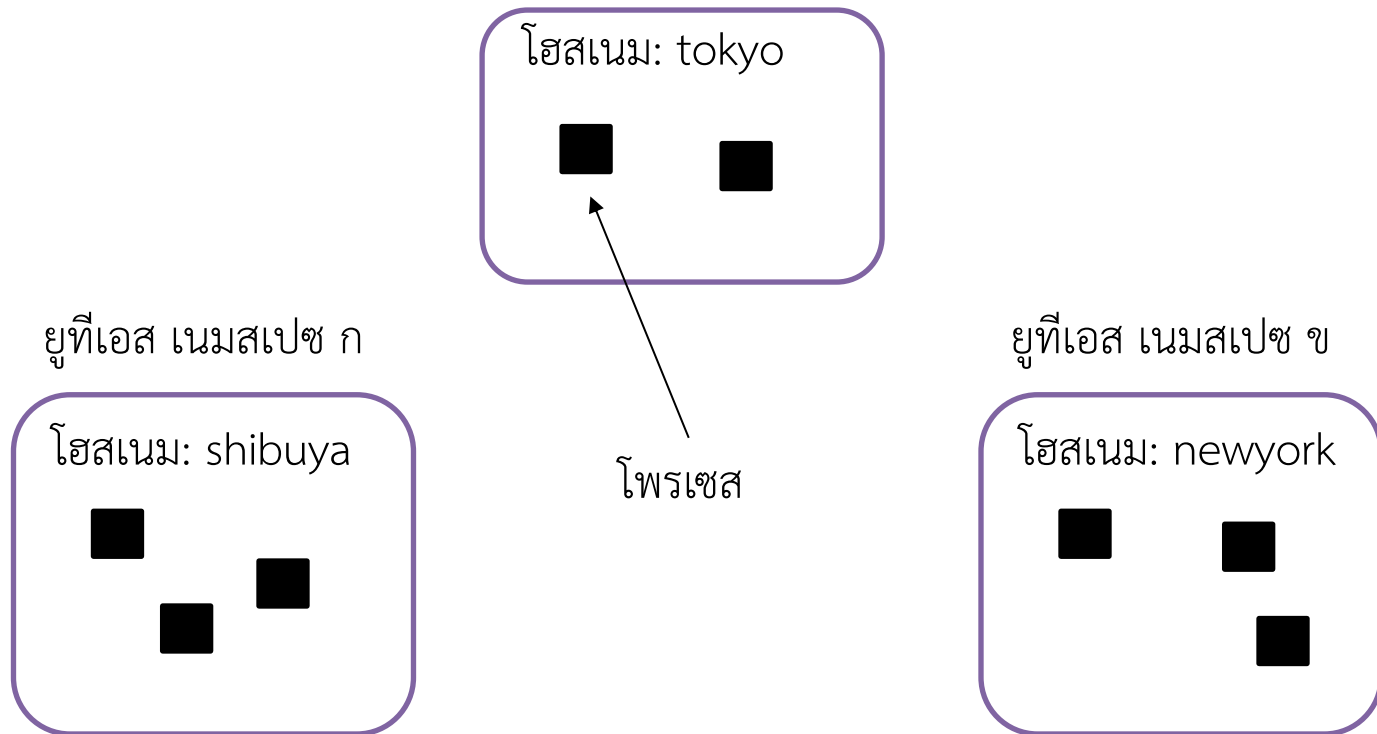
- เนมสเปซคือการสร้างโลกเสมือนให้กลุ่มของโพรเซสในเรื่องใดเรื่องหนึ่ง
ต.ย. แม้าท์เนมสเปซ แยกเซตของแม้าท์พ้อยต์ ให้เห็นเฉพาะกลุ่มของโพรเซสที่อยู่ในเนมสเปซเดียวกัน
- การใช้เนมสเปซมักจะเป็นการใช้เนมสเปซหลายชนิดไปด้วยกัน
ยกตัวอย่างเช่น แอปพลิเคชันหนึ่ง (หรือคอนเทนเนอร์หนึ่ง) อาจกำหนดให้มีกลุ่มของ พีไอดีเนมสเปซ ไอพีซีเนมสเปซ ซีกรูปเนมสเปซ และแม้าท์เนมสเปซที่ถูกใช้งานไปด้วยกัน
 - ในคอนเทนเนอร์ จะประกอบไปด้วยเซตของเนมสเปซอินสแตนซ์ที่ถูกใช้งานไปด้วยกันเพื่อสร้างภาพเสมือนของคอนเทนเนอร์ที่แยกจากคอนเทนเนอร์อื่น

ยูทียูเอสเนมสเปซ

- ยูทียูเอสย่อมาจาก UNIX Time Sharing เป็นเนมสเปซที่อนุญาตให้มีการแยกชื่อเครื่องให้เป็นหลายชื่อได้
- เมื่อใช้คำสั่ง `uname` ในยูทียูเอสเนมสเปซต่างอินสแตนซ์กันจะรีเทิร์น (1) `nodename` และ (2) `nisdomainname` ที่แตกต่างกันได้
- คอนเทนเนอร์สามารถใช้ชื่อที่แตกต่างกันเพื่อติดต่อกับ DHCP server เพื่อขอไอพีแอดเดรสแยกกันสำหรับแต่ละคอนเทนเนอร์
- โพรเซสในยูทียูเอสเนมสเปซอินสแตนซ์เดียวกันจะเห็นโฮสเนมและโดเมนเนมเดียวกัน การเปลี่ยนแปลงค่าเหล่านี้จะเห็นเฉพาะโพรเซสภายในเนมสเปซเดียวกันเท่านั้น

ยูทียูเอสเนมสเปซ

ยูทียูเอส เนมสเปซ เริ่มต้น
initial UTS namespace



	init	inst1	...	instN
uts		P1	...	P2
ipc	P1 P2		...	
net	P1 P2		...	
pid	P1		...	P2
mount	P1	P2	...	
user	P1	P2	...	
cgroup	P1 P2		...	

สร้างโปรเซสและโปรเซสลูก

```
kasidit@flyvm:~$ sh
$
$
$ $$
sh: 3: 3547: not found
$ ps
    PID TTY          TIME CMD
    3536 pts/3        00:00:00 bash
    3547 pts/3        00:00:00 sh
    3548 pts/3        00:00:00 ps
$
$ sh
$
$ $$
sh: 2: 3557: not found
$ ps
    PID TTY          TIME CMD
    3536 pts/3        00:00:00 bash
    3547 pts/3        00:00:00 sh
    3557 pts/3        00:00:00 sh
    3558 pts/3        00:00:00 ps
$
```

ดูว่าโปรเซสพ่อแม่อยู่ในเนมสเปซอินสแตนซ์ใดบ้าง

- /proc/3547/ns/*
- ตัวเลขใน `uts:[4026531838]` คือ inode # ในระบบไฟล์ซึ่ง unique

```
$ cd /proc/3547
$ ls -l ns
total 0
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 net -> 'net:[4026532000]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 user -> 'user:[4026531837]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:04 uts -> 'uts:[4026531838]'
$ readlink /proc/3547/ns/uts
uts:[4026531838]
$
```

ดูว่าโพรเซสลูกอยู่ในเนมสเปซอินสแตนซ์ใดบ้าง

- /proc/3557/ns/*

```
kasidit@flyvm:/proc/3557$ pwd
/proc/3557
kasidit@flyvm:/proc/3557$ ls -l ns
total 0
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 net -> 'net:[4026532000]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 user -> 'user:[4026531837]'
lrwxrwxrwx 1 kasidit kasidit 0 Oct 10 22:17 uts -> 'uts:[4026531838]'
kasidit@flyvm:/proc/3557$
```

เอพีไอและคำสั่ง

- ซิสเต็มคอล เอพีไอ ที่เกี่ยวข้องกับเนมสเปซ ได้แก่
 - clone(2) มีออปชั่นให้ สร้างโปรเซสลูกใน เนมสเปซใหม่ได้ และเลือกได้ด้วยว่าเนมสเปซใหม่เป็นชนิดใด
 - unshare(2) สร้างเนมสเปซใหม่ และย้ายโปรเซสที่เรียกใช้ unshare ไปอยู่ในเนมสเปซใหม่นั้น
 - setns(2) ย้ายโปรเซสที่เรียกใช้คำสั่งนี้จากเนมสเปซอินสแตนซ์ปัจจุบันไปอยู่ในอีกอินสแตนซ์หนึ่ง
- คำสั่ง
 - unshare สร้างเนมสเปซและรันคำสั่งในเนมสเปซที่สร้างขึ้น
 - nsenter เข้าสู่เนมสเปซที่มีอยู่แล้วและรันคำสั่งในเนมสเปซนั้น

```
kasidit@flyvm:/srv/kasidit/study/namespace$ unshare -h
```

Usage:

```
unshare [options] [<program> [<argument>...]]
```

Run a program with some namespaces unshared from the parent.

Options:

-m, --mount[=<file>]	unshare mounts namespace
-u, --uts[=<file>]	unshare UTS namespace (hostname etc)
-i, --ipc[=<file>]	unshare System V IPC namespace
-n, --net[=<file>]	unshare network namespace
-p, --pid[=<file>]	unshare pid namespace
-U, --user[=<file>]	unshare user namespace
-C, --cgroup[=<file>]	unshare cgroup namespace
-f, --fork	fork before launching <program>
-r, --map-root-user	map current user to root (implies --user)
--kill-child[=<signame>]	when dying, kill the forked child (implies --fork) defaults to SIGKILL
--mount-proc[=<dir>]	mount proc filesystem first (implies --mount)
--propagation slave shared private unchanged	modify mount propagation in mount namespace
--setgroups allow deny	control the setgroups syscall in user namespaces
-R, --root=<dir>	run the command with root directory set to <dir>
-w, --wd=<dir>	change working directory to <dir>
-S, --setuid <uid>	set uid in entered namespace
-G, --setgid <gid>	set gid in entered namespace

```
[kasidit@flyvm:/srv/kasidit/study/namespace$ nsenter -h
```

Usage:

```
nsenter [options] [<program> [<argument>...]]
```

Run a program with namespaces of other processes.

Options:

-a, --all	enter all namespaces
-t, --target <pid>	target process to get namespaces from
-m, --mount[=<file>]	enter mount namespace
-u, --uts[=<file>]	enter UTS namespace (hostname etc)
-i, --ipc[=<file>]	enter System V IPC namespace
-n, --net[=<file>]	enter network namespace
-p, --pid[=<file>]	enter pid namespace
-C, --cgroup[=<file>]	enter cgroup namespace
-U, --user[=<file>]	enter user namespace
-S, --setuid <uid>	set uid in entered namespace
-G, --setgid <gid>	set gid in entered namespace
--preserve-credentials	do not touch uids or gids
-r, --root[=<dir>]	set the root directory
-w, --wd[=<dir>]	set the working directory
-F, --no-fork	do not fork before exec'ing <program>
-Z, --follow-context	set SELinux context according to --target PID
-h, --help	display this help
-V, --version	display version

For more details see nsenter(1).

สิทธิ์ในการสร้างเนมสเปซ

- ถ้าสร้างอินสแตนซ์ของ ยูเซอร์เนมสเปซ ไม่ต้องใช้สิทธิ์พิเศษใด
- ถ้าสร้างอินสแตนซ์ของเนมสเปซแบบอื่น บัญชีผู้ใช้ของผู้สร้างต้องมี
CAP_SYS_ADMIN

```
kasidit@flyvm:/srv/kasidit/study/namespace$ readlink /proc/$$/ns/uts
uts:[4026531838]
kasidit@flyvm:/srv/kasidit/study/namespace$ echo $$
3630
kasidit@flyvm:/srv/kasidit/study/namespace$ sudo unshare -u bash
root@flyvm:/srv/kasidit/study/namespace# echo $$
3715
root@flyvm:/srv/kasidit/study/namespace# readlink /proc/$$/ns/uts
uts:[4026532461]
root@flyvm:/srv/kasidit/study/namespace# hostname tokyo
root@flyvm:/srv/kasidit/study/namespace# hostname
tokyo
root@flyvm:/srv/kasidit/study/namespace#
```

0 bash

```
kasidit@flyvm:/srv/kasidit/study/namespace$ readlink /proc/$$/ns/uts
uts:[4026531838]
kasidit@flyvm:/srv/kasidit/study/namespace$ echo $$
3640
kasidit@flyvm:/srv/kasidit/study/namespace$ hostname
flyvm
kasidit@flyvm:/srv/kasidit/study/namespace$ hostname
flyvm
kasidit@flyvm:/srv/kasidit/study/namespace$ █
```

```
root@flyvm:/srv/kasidit/study/namespace# readlink /proc/$$/ns/mnt
mnt:[4026531840]
root@flyvm:/srv/kasidit/study/namespace# echo $$
3715
root@flyvm:/srv/kasidit/study/namespace# hostname
tokyo
root@flyvm:/srv/kasidit/study/namespace# readlink /proc/$$/ns/uts
uts:[4026532461]
root@flyvm:/srv/kasidit/study/namespace#
```

```
0 bash
```

```
kasidit@flyvm:/srv/kasidit/study/namespace$ readlink /proc/$$/ns/mnt
mnt:[4026531840]
kasidit@flyvm:/srv/kasidit/study/namespace$ sudo nsenter -t 3715 -u bash
root@tokyo:/srv/kasidit/study/namespace#
root@tokyo:/srv/kasidit/study/namespace# readlink /proc/$$/ns/uts
uts:[4026532461]
root@tokyo:/srv/kasidit/study/namespace# hostname
tokyo
root@tokyo:/srv/kasidit/study/namespace# █
```

เนมสเปซ และ คอนเทนเนอร์

- เราใช้เนมสเปซสร้าง คอนเทนเนอร์ (container) หรือ การประมวลผลแบบเสมือนที่มีโอเวอร์เฮดน้อย (light-weighted virtualization)
- เป็นการสร้างภาพเสมือนของระบบคอมพิวเตอร์โดยการแยกกลุ่มของโปรเซส และให้โปรเซสกลุ่มนั้นคิดว่ากลุ่มของตนครอบครองทั้งระบบ
- การประมวลผลแบบเสมือนตามปกติจะแยกเกสโอเอสอย่างชัดเจนและแต่ละเกสวีเอ็มก็มีโปรเซสของมัน
 - เกสโอเอสอาจต่างชนิด (ต่างเคอร์เนล) กัน
 - มีการรักษาความปลอดภัยของเกสโอเอสแต่ละอัน
 - แต่ใช้เวลาในการรันวีเอ็มเนื่องจากเกสใช้เวลารันนาน (ยกเว้น unikernel)
- แต่การแยกกันโดยใช้เนมสเปซเป็นการแยกกันแบบที่แต่ละกลุ่มของโปรเซสถึงแม้จะไม่เห็นกันแชร์โอเอสเคอร์เนลเดียวกัน

เนมสเปซ และ คอนเทนเนอร์

- การใช้เนมสเปซมีโอเวอร์เฮดน้อยกว่าวีเอ็ม ดังนั้นจึงเริ่มต้นการประมวลผลได้เร็วกว่า
- มีความละเอียดในการกำหนดลักษณะของการแยกกันที่ละเอียดกว่า เพราะผู้ใช้สามารถเลือกได้ว่าจะแยกหรือไม่แยกเนมสเปซอินสแตนซ์ชนิดใด
- แต่ในการสร้างคอนเทนเนอร์จะต้องกำหนดรายละเอียดมากตามไปด้วย เพราะคอนเทนเนอร์หนึ่งเกี่ยวข้องกับเนมสเปซหลายแบบ และเคอร์เนลต้องมีความสามารถสนับสนุนการแยกและสร้างภาพเสมือนด้วยเนมสเปซเป็นอย่างดี และมีความซับซ้อนมาก (ใช้เวลาในการพัฒนามาก)
 - ต.ย. เนมสเปซล่าสุด ไทม์เนมสเปซ ได้ถูกเสนอมาตั้งแต่ 2018 เพิ่งอยู่ใน mainline linux kernel ในปี 2020

เนมสเปซ และ คอนเทนเนอร์

- **เนมสเปซ** จะเป็นตัวกำหนดว่ากลุ่มของโพรเซสในคอนเทนเนอร์จะเห็นอะไรในระบบได้บ้าง
- ปกติแล้วเมื่อโพรเซสสุดท้ายในเนมสเปซจบการประมวลผล เนมสเปซจะถูกทำลาย แต่ผู้ใช้สามารถใช้วิธีการ mount-binding เพื่อเก็บไฟล์ pseudo-file ของเนมสเปซนั้น เพื่อไม่ให้ถูกทำลายและนำมาใช้ภายหลังได้
- **ซีกรุป** จะเป็นตัวกำหนดว่ากลุ่มของโพรเซสในคอนเทนเนอร์มีสิทธิ์ที่จะใช้งานทรัพยากรอะไรได้บ้างและเป็นปริมาณมากน้อยขนาดไหน

เม้าท์เนมสเปซ

- แฟล็ก CLONE_NEWNS (เพราะเป็นอันแรก)
- เม้าพอยต์ ประกอบไปด้วย (1) เม้าท์ซอร์ส (อุปกรณ์/พาติชั่น) (2) พาทเนม และ(3) ไอดีของเม้าท์พาเรนท์
- มีความยืดหยุ่นกว่า chroot ซึ่งกำหนดให้โพรเซสใช้ได้เฉพาะสับทรี (subtree) ของไฟล์ซิสเต็ม แต่เม้าท์เนมสเปซอนุญาตให้เปลี่ยนแปลงได้มากกว่า
- เม้าท์เนมสเปซ อนุญาตให้ผู้ใช้สร้างภาพเสมือนของไฟล์ซิสเต็มของตนเอง สำหรับกลุ่มของโพรเซส
- เม้าท์เนมสเปซ อนุญาตให้กลุ่มของโพรเซสหรือคอนเทนเนอร์มองเห็นเม้าท์พอยต์ (private mount) ที่เห็นเฉพาะโพรเซสภายในคอนเทนเนอร์และแยกจากคอนเทนเนอร์อื่น
 - เช่นแต่ละครเทนเนอร์มี /tmp ของตนเอง

เม้าท์เนมสเปซ

- อนุญาตให้ผู้แชร์ (share) สับทรีของเม้าท์เนมสเปซหนึ่ง เรียกว่าการพรอบพาเกต (propagate) ไปยังเม้าท์เนมสเปซอื่นได้
- อนุญาตให้มีการพรอบพาเกต mount และ umount device จากเม้าท์เนมสเปซหนึ่งไปยังเม้าท์เนมสเปซอื่น
 - ต.ย. เช่น เม้า CD ในเม้าท์เนมสเปซหนึ่งแล้วพรอบพาเกตให้ทุกเม้าท์เนมสเปซ
- สามารถสร้างเม้าท์เนมสเปซพร้อมกับเนมสเปซอื่นเช่น พีไอดี ทำให้เกิด /proc/PID ขึ้นอย่างสอดคล้อง

เม้าท์เนมสเปซ

- ก่อนลินุกซ์ 2.4.19 โพรเซสทุกโพรเซสใช้เม้าท์พอยต์ของระบบเดียวกัน ร่วมกัน และซิสเต็มคอลเกี่ยวกับพาทเนมทั้งหมด ทำงานกับชุดของเม้าท์พอยต์นั้น
- ปัจจุบัน ระบบลินุกซ์จะสร้างลิสต์ของเม้าท์เนมสเปซในระบบ
 - โพรเซสแต่ละโพรเซสจะต้องอยู่ในเม้าท์เนมสเปซอินสแตนซ์อันใดอันหนึ่ง
 - เมื่อโพรเซสสุดท้ายจบการใช้งานเม้าท์เนมสเปซ เม้าท์เนมสเปซจะถูกทำลาย

ไอพีซีเนมสเปซ

- แฟล็ก CLONE_NEWIPC
- ไอพีซี (inter-process communication) ได้แก่
 - message queue
 - shared memory
 - semaphore
 - pipe เป็นต้น
- แยกไอพีซีออบเจกต์สำหรับกลุ่มของโพรเซส ออกจากของกลุ่มอื่น
- โพรเซสในกลุ่มไอพีซีเนมสเปซอินสแตนซ์เดียวกันจะเห็นไอพีซีออบเจกต์ของกลุ่ม แต่ไม่เห็นของเนมสเปซอื่น

ไอพีซีเนมสเปซ

- ภายในแต่ละเนมสเปซอินสแตนซ์ โพรเซสจะเห็น
 - system V IPC ไอดี และ
 - /dev/mqueue และ
 - อินสแตนซ์ที่เกี่ยวข้องกับไอพีซีใน /proc
 - semaphore
- ไอพีซีออปเจกต์จะถูกทำลายเมื่อลบไอพีซีเนมสเปซ

ซีก रूप नेम स पे स

- फ्लेक `CLONE_NEWCGROUP`
- गलुंखण प्रोसेस हेन स रू रूप नेम स पे स इन स्टैन्स डेव गन
- तू क व म खै ज स रू रूप
- गू फ स मीं न खण प ठ ने म तू यू न /proc/PID

เน็ตเวิร์คเนมสเปซ

- แฟล็ก CLONE_NEWNET
- แยกทรัพยากรที่เกี่ยวข้องกับเน็ตเวิร์ค เช่น ไอพีแอดเดรส และ ไร่ตังเทเบิ้ล /proc/net /sys/class/net และ netfilter กฎ firewall และ socket port numbers และ UNIX domain sockets
- ทำให้คอนเทนเนอร์มีตัวตนในระบบเครือข่าย โดยสร้างภาพเสมือนที่ทำให้คอนเทนเนอร์เหมือนกับ
 - มีอุปกรณ์เชื่อมต่อกับระบบเครือข่ายของตนเอง
 - แอปพลิเคชันที่รันอยู่ในคอนเทนเนอร์มีชุดของพอร์ตของตนเอง ที่อาจซ้ำกันกับพอร์ตในคอนเทนเนอร์อื่นได้ ทำให้คอนเทนเนอร์สามารถสร้างเอ็นพอยต์สำหรับสื่อสารกับอินเทอร์เน็ต และสามารถสร้างคอนเนคชันกับคอนเทนเนอร์อื่นและคอมพิวเตอร์อื่นบนอินเทอร์เน็ตได้
 - เช่นรันเว็บเซิร์ฟเวอร์บนพอร์ต 80 ของคอนเทนเนอร์

เน็ตเวิร์คเนมสเปซ

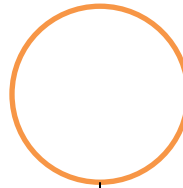
- เครื่องโฮส สามารถสร้างเราต์ติ้งเทเบิลเพื่อเราแพคเก็ตไปยังอุปกรณ์เครือข่ายเสมือนของแต่ละคอนเทนเนอร์
- อุปกรณ์ veth (virtual ethernet) เป็นอุปกรณ์ที่ใช้สร้างการเชื่อมต่อระหว่างคอนเทนเนอร์กับเครื่องโฮส
- เราสามารถย้าย เน็ตเวิร์คอินเตอร์เฟส จาก เน็ตเวิร์คเนมสเปซหนึ่งไปยังเน็ตเวิร์คเนมสเปซอื่นได้
 - `ip link set dev eth0 netns PID`

เน็ตเวิร์คเนมสเปซ

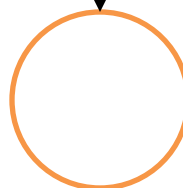
- เน็ตเวิร์คเนมสเปซทำให้คอนเทนเนอร์สามารถมีระบบเน็ตเวิร์คเสมือนของตนเองและทำตัวเป็นเซิร์ฟเวอร์ได้ และอาจใช้เป็นที่พักทดสอบและจำลองการเชื่อมต่อที่ซับซ้อนได้ด้วย แทนที่จะใช้อุปกรณ์จริง
- การแยกช่วยเรื่องความปลอดภัย แยกโปรเซสจากเน็ตเวิร์ค
 - เริ่มต้นจากเนมสเปซที่ไม่มีอุปกรณ์ใด
 - ถ้าโปรเซสที่โดนแฮกอยู่ในเน็ตเวิร์คเนมสเปซก็才不会ส่งผลกระทบกับเน็ตเวิร์คโดยรวมทั้งหมด

เน็ตเวิร์คเนมสเปซ

ไคลเอ็นท์

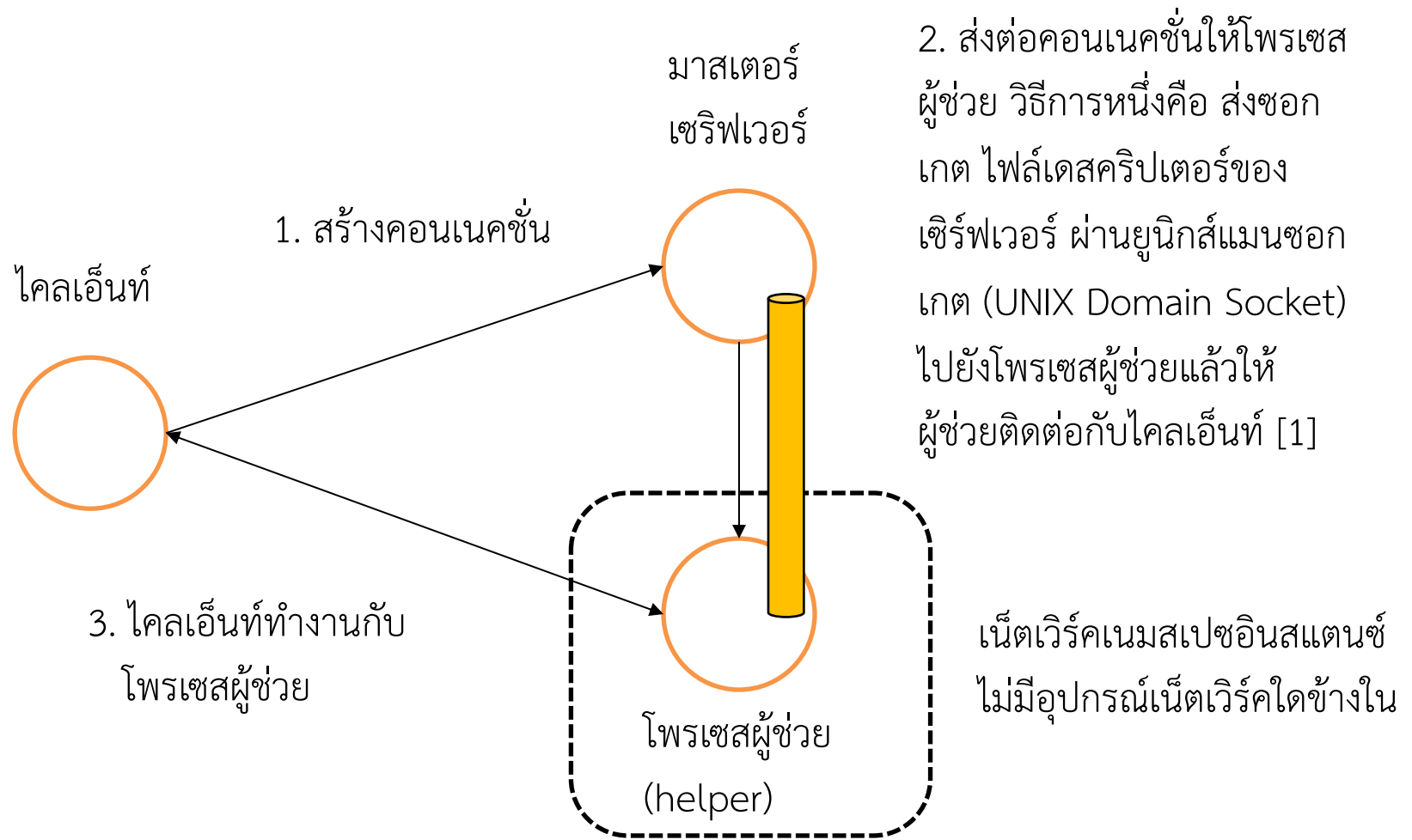


โจมตีเซิร์ฟเวอร์
โดยตรง



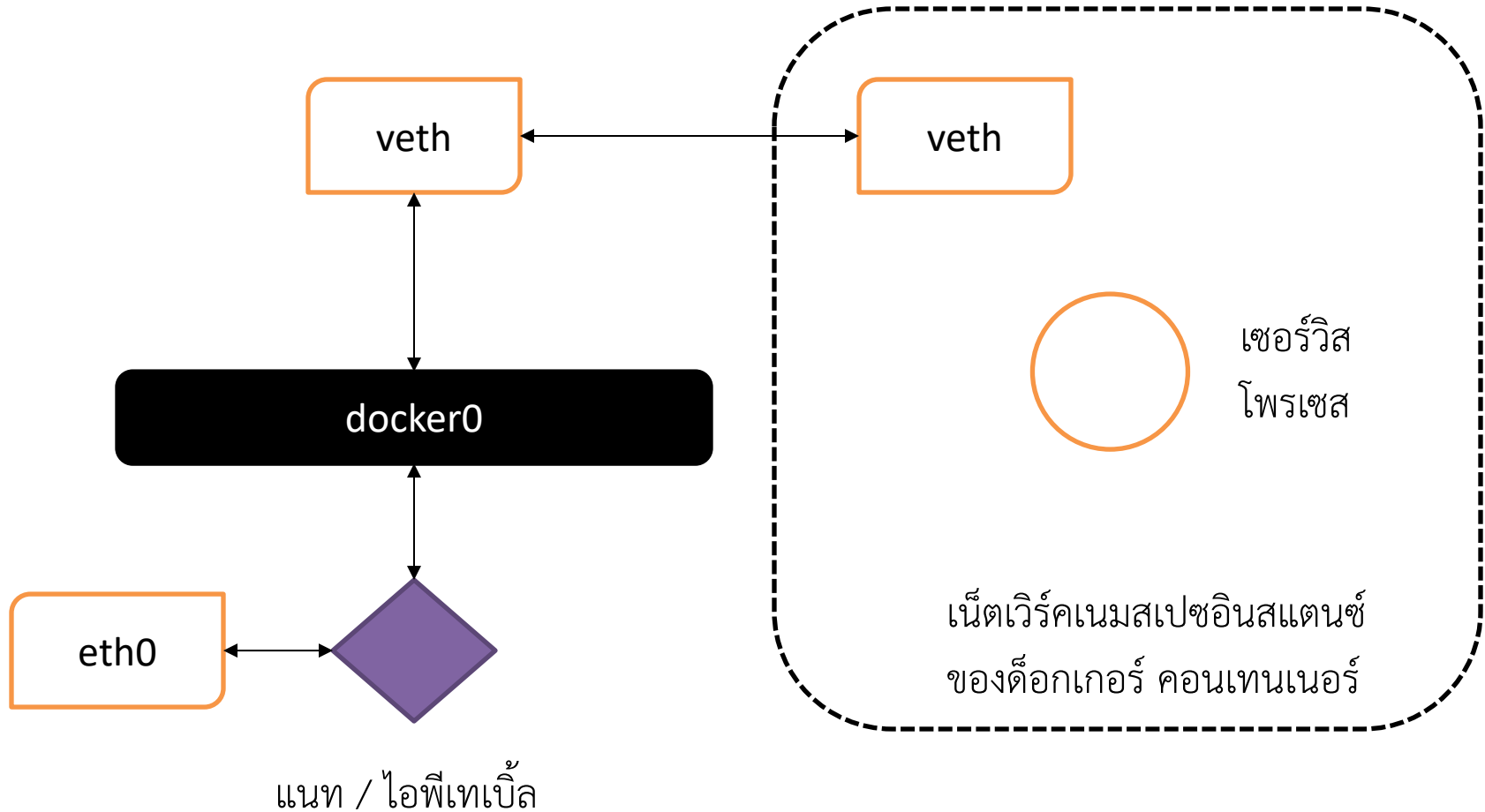
เซิร์ฟเวอร์

เน็ตเวิร์คเนมสเปซ



[1] <https://openforums.blog/2016/08/07/open-file-descriptor-passing-over-unix-domain-sockets/>

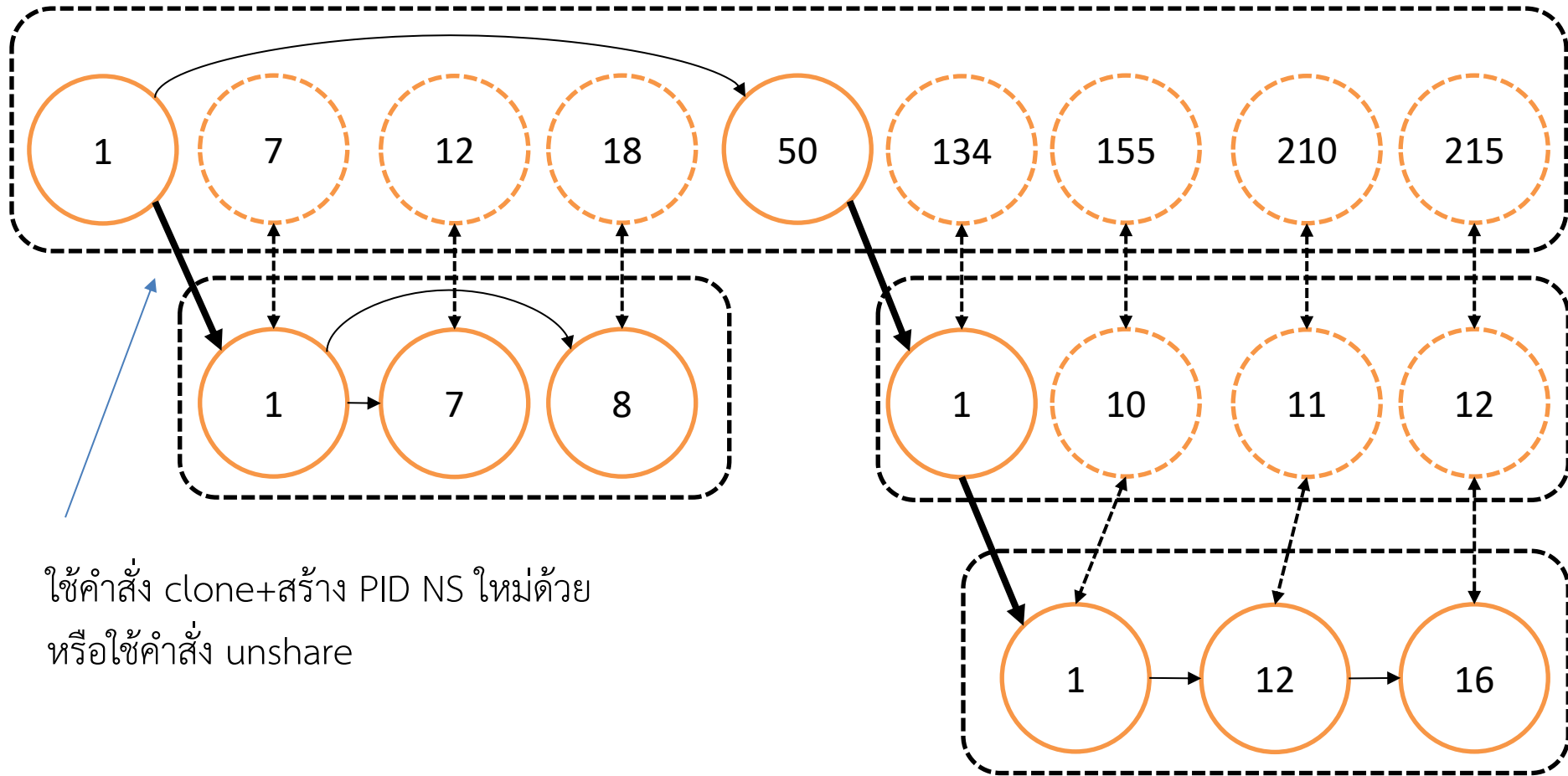
เน็ตเวิร์คเนมสเปซ ใน ด็อกเกอร์



พีไอดีเนมสเปซ

- แฟล็ก CLONE_NEWPID
- แยกพีไอดีของโพรเซสในคอนเทนเนอร์ ทำให้สามารถไม่เกรตคอนเทนเนอร์ไปรันบนโฮสเครื่องใหม่ได้ โดยไม่ต้องกลัวว่าพีไอดีจะซ้ำ
- ความสัมพันธ์ระหว่างพีไอดีเนมสเปซอินสแตนซ์เป็นแบบแผนภาพต้นไม้ เริ่มต้นจากอินสแตนซ์เริ่มต้น
- พีไอดีเนมสเปซอนุญาตให้แต่ละพีไอดีเนมสเปซอินสแตนซ์มีโพรเซส init ซึ่งมีพีไอดี 1 และจะเป็นพารেন্টของโพรเซสที่เป็นลูกกำพร้าในพีไอดีเนมสเปซอินสแตนซ์ และเราสามารถใช้โพรเซสพีไอดี 1 เพื่อกำหนดค่าเริ่มต้น
- พารেন্টพีไอดีอินสแตนซ์จะเห็นพีไอดีของโพรเซสทุกโพรเซสที่อยู่ในอินสแตนซ์ลูกและอินสแตนซ์ในทุกระดับที่เกิดจากมัน อินสแตนซ์ลูกจะไม่เห็นโพรเซสของอินสแตนซ์พ่อแม่
- พีไอดีอินสแตนซ์ในลินุกซ์มี 32 ระดับ

พีไอดีเนมสเปซ



ใช้คำสั่ง clone+สร้าง PID NS ใหม่ด้วย
หรือใช้คำสั่ง unshare

พีไอดีเนมสเปซ

- โพรเซสยังอยู่ในเนมสเปซอินสแตนซ์เดียว แต่โพรเซสในเนมสเปซของบรรพบุรุษแค่เห็นพีไอดีที่เป็นตัวแทนของโพรเซสในพีไอดีอินสแตนซ์ระดับทายาทของมันเท่านั้น
 - ถ้าต้องการ พาเรนทเนมสเปซจะเห็น เทรดไอดีของโพรเซสที่มันมองเห็นได้โดยดูจาก /proc
- เมื่อเรียก getpid() ในพีไอดีเนมสเปซ จะได้ค่าพีไอดีในพีไอดีเนมสเปซที่โพรเซสนั้นอยู่
- ถ้าพีไอดี 1 ในพีไอดีเนมสเปซอินสแตนซ์เรียก getppid() จะได้ค่า 0 เพราะมองไม่เห็นพาเรนท

พีไอดีเนมสเปซ

- เพื่อให้พีไอดีเนมสเปซอินสแตนซ์ มี `/proc/PID` ที่ถูกต้องจะต้องสร้างเมาท์เนมสเปซอินสแตนซ์ด้วย และ
- `mount /proc` สำหรับพีไอดีอินสแตนซ์นั้น จึงจะทำให้ได้ค่า `/proc/PID` ที่ถูกต้องซึ่งจำเป็นสำหรับซอฟต์แวร์เช่น `ps` และ `top`
- ถ้าโพรเซสที่มีพีไอดี 1 ใน พีไอดีเนมสเปซจบการทำงาน โพรเซสในพีไอดีเนมสเปซทั้งหมดจะจบการทำงานและ พีไอดีเนมสเปซอินสแตนซ์จะจบลง
- เหมาะกับการจบการประมวลผลของคอนเทนเนอร์

```
$ sudo unshare -p -f bash
root@flyvm:/srv/kasidit/study/namespace# echo $$
1
# ps
  PID TTY          TIME CMD
 3926 pts/1    00:00:00 sudo
 3927 pts/1    00:00:00 unshare
 3928 pts/1    00:00:00 bash
 3935 pts/1    00:00:00 ps
# ls /proc
1      1227  1406  247   2728  3055  3265  3649  46    830          execdomains  pagetypeinfo
10     123   1407  248   28     3061  3288  3659  463   831          fb           partitions
...
# less /proc/1/status
```

```
$ sudo unshare -p -m -f bash
#
# echo $$
1
# mount -t proc none /proc
# ls /proc
1      cpuinfo      fs           kmsg         modules      scsi          thread-self
9      crypto      interrupts   kpagecgroup mounts        self          timer_list
...
# ps
  PID TTY          TIME CMD
    1 pts/1    00:00:00 bash
   10 pts/1    00:00:00 ps
# less /proc/1/status
```

ยูเซอร์เนมสเปซ

- แฟล็ก CLONE_NEWUSER
- แยกผู้ใช้และกรุปไอดีของผู้ใช้
- ในลิสต์ข้อมูลของโปรเซส ค่า ยูไอดี (UID) และจีไอดี (GID) ของโปรเซสที่มองจากภายในยูเซอร์เนมสเปซ และจากภายนอกยูเซอร์เนมสเปซสามารถต่างกันได้
- ภายนอกยูเซอร์เนมสเปซอินสแตนซ์ โปรเซสมี ยูไอดีปกติ แต่ภายในยูเซอร์เนมสเปซอินสแตนซ์ ของโปรเซสนั้น โปรเซสอาจมียูไอดีเป็น 0 คือเป็น superuser สำหรับการประมวลผลภายในเนมสเปซ
- การสร้างยูเซอร์เนมสเปซอินสแตนซ์ไม่จำเป็นต้องใช้ sudo แต่พอสร้างแล้ว โปรเซสในเนมสเปซสามารถรันเป็นรูทได้ (ภายในขอบเขตของเนมสเปซ)

ยูเซอร်เนมสเปซ

- อนุญาตให้ผู้ใ้แมป ช่วงของ UID ของกลุ่มของโพรเซสในยูเซอร်เนมสเปซ กับ UID ของเครื่องจริง เช่น
 - ในคอนเทนเนอร์ A UID 0 ถึง 999 จริงๆแล้วคือ UID 2000 ถึง 2999 บนเครื่องจริง
 - ในคอนเทนเนอร์ B UID 0 ถึง 999 จริงๆแล้วคือ UID 3000 ถึง 3999 บนเครื่องจริง
- ทำโพรเซสของ UID 0 ในคอนเทนเนอร์สามารถปฏิบัติงานเป็นรูลของคอนเทนเนอร์นั้นได้

สรุป

- แนะนำ Namespaces
- ชนิดของ Namespaces
- แนะนำ Namespace แต่ละชนิดที่เกี่ยวข้องกับคอนเทนเนอร์
- ต่อไปเราจะแนะนำ ซีกรูป