**การบริหารจัดการบริการ**

Daemon Process

โปรแกรม/คำสั่งเมื่อดำเนินการ ระบบจัดเตรียมอินสแตนซ์พิเศษให้กับกระบวนการ อินสแตนซ์นี้ประกอบด้วยบริการ/ทรัพยากรทั้งหมดที่อาจใช้โดยกระบวนการภายใต้การดำเนินการ

เมื่อใดก็ตามที่ออกคำสั่งใน Unix/Linux คำสั่งนั้นจะสร้าง/เริ่มกระบวนการใหม่ ตัวอย่างเช่น เมื่อออก pwd ซึ่งใช้เพื่อแสดงตำแหน่งไดเร็กทอรีปัจจุบันที่ผู้ใช้อยู่ กระบวนการจะเริ่มต้นขึ้น

ด้วยหมายเลข ID 5 หลัก Unix/Linux จะเก็บบัญชีของกระบวนการ หมายเลขนี้คือ Call Process ID หรือ PID แต่ละกระบวนการในระบบมี PID ที่ไม่ซ้ำกัน

pid ที่ใช้หมดแล้วสามารถนำมาใช้ได้อีกครั้งสำหรับกระบวนการที่ใหม่กว่าเนื่องจากมีการใช้ชุดค่าผสมที่เป็นไปได้ทั้งหมด

ณ เวลาใดก็ตาม ไม่มีสองกระบวนการที่มี pid เดียวกันอยู่ในระบบ เนื่องจากเป็น pid ที่ Unix ใช้เพื่อติดตามแต่ละกระบวนการ

สำหรับโปรแกรมที่กำลังทำงานอยู่ (ชื่อกระบวนการ) Pidof ค้นหารหัสกระบวนการ (pids)

ฟิลด์ที่อธิบายโดย ps อธิบายว่า:

UID: User ID ที่เป็นของกระบวนการนี้ (บุคคลที่เรียกใช้)

PID: รหัสกระบวนการ

PPID: รหัสกระบวนการหลัก (รหัสของกระบวนการที่เริ่มต้น)

C: การใช้ CPU ของกระบวนการ

STIME: เวลาเริ่มต้นของกระบวนการ

TTY: ประเภทเทอร์มินัลที่เกี่ยวข้องกับกระบวนการ

เวลา: กระบวนการใช้เวลา CPU

CMD: คำสั่งที่เริ่มต้นกระบวนการนี้

มีตัวเลือกอื่นที่สามารถใช้ร่วมกับคำสั่ง ps :

-a: แสดงข้อมูลเกี่ยวกับผู้ใช้ทั้งหมด

-x: แสดงข้อมูลเกี่ยวกับกระบวนการที่ไม่มีเทอร์มินัล

-u: แสดงข้อมูลเพิ่มเติมเช่นตัวเลือก -f

-e: แสดงข้อมูลเพิ่มเติม

เริ่มต้นกระบวนการ

กระบวนการสามารถดำเนินการได้สองวิธี:

วิธีที่ 1: กระบวนการเบื้องหน้า :ทุกกระบวนการเมื่อเริ่มทำงานในเบื้องหน้าตามค่าเริ่มต้น รับอินพุตจากแป้นพิมพ์ และส่งเอาต์พุตไปยังหน้าจอ เมื่อออกคำสั่ง pwd

$ ls pwd

Output:

$ /home/geeksforgeeks/root

เมื่อคำสั่ง/กระบวนการทำงานในพื้นหน้าและใช้เวลานาน จะไม่สามารถเรียกใช้หรือเริ่มกระบวนการอื่นได้ เนื่องจากพรอมต์จะไม่พร้อมใช้งานจนกว่าโปรแกรมจะเสร็จสิ้นการประมวลผลและออกมา

วิธีที่ 2: กระบวนการพื้นหลัง:ทำงานในพื้นหลังโดยไม่ต้องป้อนข้อมูลด้วยแป้นพิมพ์และรอจนกว่าจะจำเป็นต้องป้อนข้อมูลด้วยแป้นพิมพ์ ดังนั้น กระบวนการอื่นๆ สามารถทำควบคู่ไปกับกระบวนการที่ทำงานอยู่เบื้องหลังได้ เนื่องจากไม่ต้องรอให้กระบวนการก่อนหน้านี้เสร็จสิ้น

การเพิ่ม & พร้อมกับคำสั่งเริ่มต้นเป็นกระบวนการพื้นหลัง

$ pwd &

เนื่องจาก pwd ไม่ต้องการอินพุตใด ๆ จากแป้นพิมพ์ มันจะไปที่สถานะหยุดจนกว่าจะย้ายไปยังส่วนหน้าและให้ข้อมูลใด ๆ ดังนั้นเมื่อกด Enter:

เอาต์พุต:

[1] + Done pwd

$

บรรทัดแรกนั้นมีข้อมูลเกี่ยวกับกระบวนการเบื้องหลัง – หมายเลขงานและรหัสกระบวนการ มันบอกคุณว่ากระบวนการพื้นหลังของคำสั่ง ls เสร็จสิ้นเรียบร้อยแล้ว ประการที่สองคือพรอมต์สำหรับคำสั่งอื่น

**Startup and Shutdown Script**

ใช้ขั้นตอนนี้เป็นวิธีการที่สอดคล้องกันในการเริ่มบริการบน Unix ข้อดีด้านหนึ่งของขั้นตอนนี้คือป้องกันไม่ให้แอปพลิเคชันพื้นที่ผู้ใช้ออกจากกระบวนการเริ่มต้น/ปิดระบบ ดังนั้นจึงไม่กีดขวางระบบระหว่างการบู๊ตหรือปิดระบบ

โครงสร้างไดเร็กทอรีระบบ "/usr/app/" มีอยู่เพื่อให้ความสามารถในการเริ่มต้น/ปิดระบบนี้ แต่ละบริการจะมีไดเร็กทอรีย่อยของตนเองซึ่งมีสคริปต์ที่จำเป็นในการเริ่ม หยุด และดำเนินการล้างข้อมูลรายวัน บริการถูกลงทะเบียนในไฟล์ "/usr/app/services.node" ดังนั้นระบบจึงรู้ว่าจะเริ่ม/หยุดบริการใด

ฟังก์ชัน "การล้างข้อมูล" พื้นฐานมีให้หากคุณสร้างสคริปต์ "rc.cleanup" สคริปต์นี้จะทำงานทุกวันเวลา 04:15 น. การใช้งานทั่วไปรวมถึงการล้างไฟล์บันทึกและ/หรือไฟล์ชั่วคราว

อย่าติดตั้งไฟล์เฉพาะแอปพลิเคชันอื่นๆ ใน "/usr/app/" ไดเร็กทอรีนั้นควรมีเพียงสามสคริปต์สำหรับการเริ่มต้น (rc.local), การปิดระบบ (rc.shutdown) และสคริปต์การล้างข้อมูลที่เป็นทางเลือก (rc.cleanup)

สร้างไดเรกทอรีย่อยของ /usr/app สำหรับบริการของคุณ: sudo mkdir /usr/app/SERVICENAME

สร้างไดเร็กทอรีย่อย "etc" เพื่อเก็บสคริปต์ของคุณ: sudo mkdir /usr/app/SERVICENAME/etc

สร้างสคริปต์เริ่มต้น "rc.local": sudo vi /usr/app/SERVICENAME/etc/rc.local

สร้างสคริปต์ปิดระบบ "rc.shutdown": sudo vi /usr/app/SERVICENAME/etc/rc.shutdown

ทางเลือก: สร้างสคริปต์การล้างข้อมูลรายวัน "rc.cleanup":

sudo vi /usr/app/SERVICENAME/etc/rc.cleanup

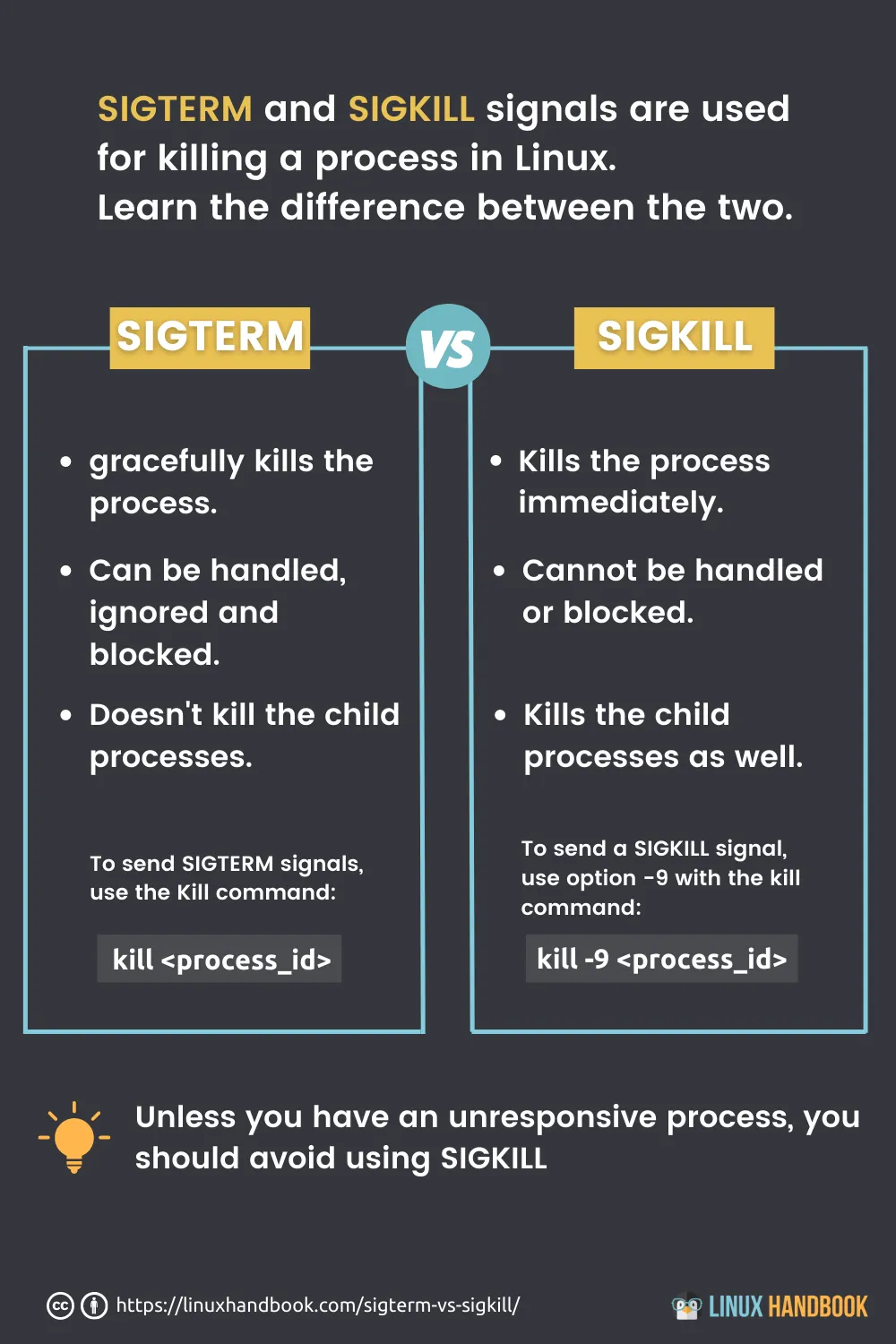
ทดสอบสคริปต์ของคุณด้วยตนเองเพื่อให้แน่ใจว่าสคริปต์เริ่มต้น ปิด และล้างข้อมูลจริงหลังจากแอปพลิเคชันของคุณ

เพิ่มชื่อไดเร็กทอรีของคุณใน "services.node": sudo vi /usr/app/services.node

ชื่อไดเร็กทอรีสัมพันธ์กับ "/usr/app/" และไม่ควรมีเครื่องหมายทับ

ดำเนินการต่อจากตัวอย่าง เพิ่ม SERVICENAME ลงในไฟล์ทีละบรรทัด

SIGTERM vs SIGKULL vs SIGHUP



SIGTERM คืออะไร?

ในระบบที่คล้าย UNIX สัญญาณ SIGTERM ใช้สำหรับยุติโปรแกรม คุณสามารถคาดเดาได้จากชื่อของมัน ซึ่งประกอบด้วย SIGnal และ TERMinate

SIGTERM ยังสามารถอ้างถึงเป็น soft kill เนื่องจากกระบวนการที่รับสัญญาณ SIGTERM อาจเลือกที่จะเพิกเฉย

**จะส่ง SIGTERM ไปยังกระบวนการใน Linux ได้อย่างไร**

คำสั่ง kill ใน Linux ใช้สำหรับส่งสัญญาณดังกล่าวทั้งหมดไปยังกระบวนการ

ตามค่าเริ่มต้น คำสั่ง kill จะส่งสัญญาณ SIGTERM คุณอาจพูดถึงอย่างชัดเจนด้วย -15 แต่นั่นซ้ำซ้อน

คุณจะต้องทราบ pid ของกระบวนการเพื่อใช้คำสั่งนี้ในลักษณะต่อไปนี้:

kill <process\_id>

คุณสามารถใช้คำสั่ง ps ใน Linuxเพื่อรับ ID กระบวนการ

SIGKILL คืออะไร?

SIGKILL ใช้สำหรับยุติกระบวนการในทันที สัญญาณนี้ไม่สามารถเพิกเฉยหรือปิดกั้นได้ กระบวนการจะสิ้นสุดลงพร้อมกับเธรด (ถ้ามี)

เป็นวิธีที่โหดร้ายในการฆ่ากระบวนการ และควรใช้เป็นทางเลือกสุดท้ายเท่านั้น สมมติว่าคุณมีกระบวนการที่ไม่ตอบสนองซึ่งคุณต้องการปิด SIGKILL สามารถใช้ในกรณีดังกล่าวได้

**จะส่ง SIGKILL ไปยังกระบวนการใน Linux ได้อย่างไร**

คุณสามารถใช้ตัวเลือก -9 เพื่อส่งสัญญาณ SIGKILL ด้วยคำสั่ง kill และฆ่ากระบวนการทันที:

kill -9 <process\_id>

**SIGTERM กับ SIGKILL: เหตุใดคุณจึงควรใช้ SIGTERM มากกว่า SIGKILL**

แม้ว่าสัญญาณทั้งสองนี้จะใช้สำหรับการฆ่ากระบวนการ แต่ก็มีข้อแตกต่างระหว่างสองอย่างนี้:

SIGTERM ฆ่ากระบวนการอย่างสง่างามในขณะที่ SIGKILL ฆ่ากระบวนการทันที

สัญญาณ SIGTERM สามารถจัดการ เพิกเฉย และบล็อกได้ แต่ไม่สามารถจัดการหรือบล็อก SIGKILL ได้

SIGTERM ไม่ฆ่ากระบวนการลูก SIGKILL ฆ่ากระบวนการลูกเช่นกัน

ผู้ใช้ Linux บางคนติดนิสัยใช้ 'kill -9' และนั่นคือสิ่งที่คุณควรหลีกเลี่ยง คุณไม่จำเป็นต้องใช้ SIGKILL เว้นแต่คุณจะมีกระบวนการที่ไม่ตอบสนอง

ด้วย SIGTERM โปรเซสจะได้รับเวลาในการส่งข้อมูลไปยังพาเรนต์และโปรเซสย่อย กระบวนการย่อยถูกจัดการโดย init

การใช้ SIGKILL อาจนำไปสู่การสร้างกระบวนการซอมบี้ เนื่องจากกระบวนการที่ถูกฆ่าไม่มีโอกาสที่จะบอกกระบวนการหลักว่าได้รับสัญญาณการฆ่า

SIGHUP คืออะไร?

SIGHUP, HUP – สัญญาณนี้บอกกระบวนการว่าเซสชันของผู้ใช้สิ้นสุดลงแล้ว และควรยุติกระบวนการ

ขึ้นอยู่กับการดำเนินการของกระบวนการ ซึ่งอาจทำให้กระบวนการเริ่มต้นใหม่แทน โดยทั่วไป ลักษณะการทำงานนี้จะพบได้ในโปรแกรมที่ไม่ใช่แบบโต้ตอบซึ่งเซสชันของผู้ใช้ไม่มีความหมาย

ตัวอย่างการใช้สัญญาณ “SIGHUP” สำหรับคำสั่ง kill

สำหรับตัวอย่างนี้ เราจะใช้คำสั่ง kill เพื่อส่งสัญญาณ “SIGHUP” บนโปรเซสเซอร์ที่มี ID “28367“

คุณสามารถระบุสัญญาณ SIGHUP ได้โดยใช้ “1“, “SIGHUP” หรือ “HUP” หลังตัวเลือก “-s”

ด้านล่างนี้ คุณสามารถดูคำสั่งสามคำสั่งที่จะส่งสัญญาณ SIGHUP เดียวกันไปยังกระบวนการที่เราระบุ แต่ใช้รูปแบบที่แตกต่างกันเล็กน้อย

คำสั่งแรกนี้ใช้ชื่อเต็มของสัญญาณ

kill -s SIGHUP 28367 หรือ kill -s HUP 28367

Crontab Concepts and Further

crontab คือรายการ คำสั่งที่คุณต้องการเรียกใช้ตามกำหนดเวลาปกติ และยังเป็นชื่อของคำสั่งที่ใช้จัดการรายการนั้นด้วย Crontab ย่อมาจาก "cron table" เนื่องจากใช้cron ตัวกำหนดตารางเวลางาน เพื่อดำเนินงาน cronนั้นได้รับการตั้งชื่อตาม "chronos" ซึ่งเป็นคำในภาษากรีกที่แปลว่าเวลา cronคือกระบวนการของระบบที่จะทำงานให้คุณโดยอัตโนมัติตามกำหนดเวลาที่ตั้งไว้ กำหนดการเรียกว่า crontab ซึ่งเป็นชื่อของโปรแกรมที่ใช้ในการแก้ไขกำหนดการนั้น รูปแบบ Linux Crontab

MIN HOUR DOM MON DOW CMD

ฟิลด์ Crontab และช่วงที่อนุญาต (Linux Crontab Syntax)

Field Description Allowed Value

MIN Minute field 0 to 59

HOUR Hour field 0 to 23

DOM Day of Month 1-31

MON Month field 1-12

DOW Day Of Week 0-6

CMD Command Any command to be executed.

ตัวอย่างของงาน Cron 1. การจัดตารางงานตามเวลาที่กำหนด การใช้งานพื้นฐานของ cron คือการสั่งงานให้เสร็จภายในเวลาที่กำหนด ดังภาพด้านล่าง สิ่งนี้จะรันสคริปต์เชลล์สำรองข้อมูลทั้งหมด (สำรองข้อมูลทั้งหมด) ในวันที่ 10 มิถุนายน 08:30 น. ช่องเวลาใช้รูปแบบ 24 ชั่วโมง ดังนั้นสำหรับ 8.00 น. ให้ใช้ 8 และสำหรับ 20.00 น. ให้ใช้ 20

30 08 10 06 \* /home/maverick/full-backup >>> 30 – 30 นาที 08 – 08 น. 10 – 10 วันที่ 06 – 6 เดือน (มิถุนายน) \* – ทุกวันในสัปดาห์2. หากต้องการดูรายการ Crontab

ดูรายการ Crontab ของผู้ใช้ที่เข้าสู่ระบบปัจจุบัน : หากต้องการดูรายการ crontab ของคุณ ให้พิมพ์ crontab -l จากบัญชียูนิกซ์ของคุณ

Text

Description automatically generated

ดูรายการ Root Crontab : เข้าสู่ระบบในฐานะผู้ใช้รูท (su – root) และทำ crontab -l

A screenshot of a computer

Description automatically generated with medium confidence

หากต้องการดูรายการ crontab ของผู้ใช้ Linux รายอื่น: เข้าสู่ระบบรูทและใช้ -u {ชื่อผู้ใช้} -l

Text

Description automatically generated

3.หากต้องการแก้ไขรายการ Crontab แก้ไขรายการ Crontabของผู้ใช้ที่เข้าสู่ระบบปัจจุบัน หากต้องการแก้ไขรายการ crontab ให้ใช้ crontab -e โดยค่าเริ่มต้น การดำเนินการนี้จะแก้ไข crontab ของผู้ใช้ที่เข้าสู่ระบบในปัจจุบัน

A screenshot of a computer

Description automatically generated with medium confidence

4.จัดตารางงานทุกนาทีโดยใช้ Cron ตามหลักการแล้ว คุณอาจไม่จำเป็นต้องกำหนดเวลางานทุกนาที แต่การทำความเข้าใจตัวอย่างนี้จะช่วยให้คุณเข้าใจตัวอย่างอื่นๆ

\* \* \* \* \* CMD

\* หมายถึงหน่วยที่เป็นไปได้ทั้งหมด — เช่น ทุกนาที ทุกชั่วโมงตลอดทั้งปี มากกว่าการใช้ \* นี้โดยตรง คุณจะพบว่ามีประโยชน์มากในกรณีต่อไปนี้ เมื่อคุณระบุ \*/5 ในฟิลด์นาที หมายถึงทุกๆ 5 นาที เมื่อคุณระบุ 0-10/2 ในฟิลด์นาที หมายถึงทุกๆ 2 นาทีใน 10 นาทีแรก ดังนั้นข้อตกลงข้างต้นสามารถใช้กับฟิลด์อื่น ๆ ทั้งหมด 4 ฟิลด์

5.จัดตารางงานมากกว่าหนึ่งครั้ง (เช่น สองครั้งต่อวัน)สคริปต์ต่อไปนี้จะทำการสำรองข้อมูลส่วนเพิ่มวันละสองครั้งทุกวัน ตัวอย่างนี้รันสคริปต์เชลล์สำรองส่วนเพิ่มที่ระบุ (สำรองส่วนเพิ่ม) เวลา 11:00 น. และ 16:00 น. ของทุกวัน ค่าที่คั่นด้วยเครื่องหมายจุลภาคในฟิลด์ระบุว่าคำสั่งจำเป็นต้องดำเนินการตลอดเวลาที่กล่าวถึง

00 11, 16 \* \* \* /home/maverick/bin/incremental-backup

00 – นาทีที่ 0 (บนสุดของชั่วโมง) 11, 16 – 11.00 น. และ 16.00 น. \* – ทุกวัน \* – ทุกเดือน \* – ทุกวันในสัปดาห์ )หากคุณต้องการให้งานถูกกำหนดเวลาทุกชั่วโมงในช่วงเวลาที่กำหนด ให้ใช้สิ่งต่อไปนี้

Cron Job ทุกวันในช่วงเวลาทำงาน : ตัวอย่างนี้ตรวจสอบสถานะของฐานข้อมูลทุกวัน (รวมวันหยุดสุด

สัปดาห์) ในช่วงเวลาทำงาน 9.00 น. – 18.00 น.

00 09-18 \* \* \* /home/maverick/bin/check-db-status

00 – นาทีที่ 0 (บนสุดของชั่วโมง) 09-18 – 9.00 น. 10.00 น. 11.00 น. 12.00 น. 13.00 น. 14.00 น. 15.00 น. 16.00 น. 17.00 น. 18.00 น. \* – ทุกวัน \* – ทุกวัน เดือน \* – ทุกวันในสัปดาห์

Cron Job ทุกวันธรรมดาในช่วงเวลาทำงาน : ตัวอย่างนี้ตรวจสอบสถานะของฐานข้อมูลทุกวันทำงาน (เช่น ไม่รวมวันเสาร์และอาทิตย์) ในช่วงเวลาทำงาน 9.00 น. – 18.00 น.

00 09-18 \* \* 1-5 /home/maverick/bin/check-db-status

00 – นาทีที่ 0 (บนสุดของชั่วโมง) 09-18 – 9.00 น. 10.00 น. 11.00 น. 12.00 น. 13.00 น. 14.00 น. 15.00 น. 16.00 น. 17.00 น. 18.00 น. \* – ทุกวัน \* – ทุกวัน เดือนที่ 1-5 - จันทร์ อังคาร พุธ พฤหัสบดี และศุกร์ (ทุกวันธรรมดา)

7.กำหนดการทำงาน Cron ในเบื้องหลังทุกๆ 10 นาที ใช้สิ่งต่อไปนี้ หากคุณต้องการตรวจสอบพื้นที่ดิสก์ทุกๆ 10 นาที

\*/10 \* \* \* \* /home/maverick/check-disk-space

มันดำเนินการคำสั่งที่ระบุตรวจสอบพื้นที่ดิสก์ทุก ๆ 10 นาทีตลอดทั้งปี แต่คุณอาจมีข้อกำหนดในการดำเนินการคำสั่งเฉพาะในบางช่วงเวลาหรือในทางกลับกัน ตัวอย่างข้างต้นแสดงวิธีการทำสิ่งเหล่านั้น แทนที่จะระบุค่าใน 5 ช่อง เราสามารถระบุโดยใช้คำสำคัญคำเดียวตามที่กล่าวไว้ด้านล่าง มีกรณีพิเศษที่คุณสามารถใช้ @ ตามด้วยคีย์เวิร์ดแทนฟิลด์ 5 ฟิลด์ด้านบน เช่น รีบูต เที่ยงคืน รายปี รายชั่วโมง คำหลักพิเศษของ Cron และความหมาย

Keyword Equivalent

@yearly 0 0 1 1 \*

@daily 0 0 \* \* \*

@hourly 0 \* \* \* \*

@reboot Run at startup.

8. กำหนดเวลางานในนาทีแรกของทุกปีโดยใช้ @yearlyหากคุณต้องการให้งานทำงานในนาทีแรกของทุกปี คุณสามารถใช้คีย์เวิร์ด @yearly cron ดังที่แสดงด้านล่าง ซึ่งจะทำให้ระบบทำงานเป็นรายปี การบำรุงรักษาโดยใช้เชลล์สคริปต์การบำรุงรักษาประจำปีในเวลา 00:00 น. ของวันที่ 1 มกราคมของทุกปี

@yearly /home/maverick/bin/annual-maintenance

9.จัดตารางงาน Cron ทุกต้นเดือนโดยใช้ @monthlyซึ่งคล้ายกับ @yearly ข้างต้น แต่รันคำสั่งทุกเดือนโดยใช้คีย์เวิร์ด @monthly cron ซึ่งจะดำเนินการสำรองเทปเชลล์สคริปต์เวลา 00:00 น. ของวันที่ 1 ของทุกเดือน

@monthly /home/maverick/bin/tape-backup

10.เพื่อจัดตารางงานพื้นหลังทุกวันโดยใช้ @dailyการใช้คีย์เวิร์ด @daily cron จะเป็นการล้างไฟล์บันทึกรายวันโดยใช้สคริปต์เชลล์ cleanup-logs เวลา 00:00 น. ของทุกวัน

@daily /home/maverick/bin/cleanup-logs "day started"

11.ในการดำเนินการคำสั่ง linux ทุกครั้งหลังจากรีบูตโดยใช้ @rebootการใช้คีย์เวิร์ด @reboot cron จะเป็นการดำเนินการตามคำสั่งที่ระบุหลังจากบูตเครื่องทุกครั้ง

@reboot CMD

**การบริหารจัดการผู้ใช้สิทธิ**

**/ect/passwd & /etc/shadow concepts and Further**

/etc/passwd มีข้อมูลผู้ใช้ทั้งหมด ไม่มีเงา อย่างไรก็ตาม พบว่ามีการโจมตีพจนานุกรมในไฟล์เพื่อค้นหารหัสผ่าน (หากอยู่ในพจนานุกรม)

ดังนั้นจึงตัดสินใจลบรหัสผ่านออกจาก /etc/passwd ไฟล์ที่เหลือยังคงอยู่ เนื่องจากมีการใช้โปรแกรมมากมายเช่น ล. รหัสผ่านถูกย้ายไปที่ /etc/shadow และไฟล์นี้ถูกสร้างขึ้นเพื่อให้รูทเท่านั้นที่สามารถอ่านได้

/etc/passwd มี x สำหรับฟิลด์รหัสผ่าน /etc/shadow แชร์เฉพาะฟิลด์แรก (ฟิลด์คีย์ / ชื่อผู้ใช้) /etc/shadow ถูกขยายให้มีช่องจัดการรหัสผ่านอื่นๆ

Su & sudo Configuration with Examples

sudo vs su

ทั้ง su และ sudo คือการยกระดับสิทธิ์ที่กำหนดให้กับผู้ใช้ปัจจุบัน

ข้อแตกต่างหลักระหว่างทั้งสองคือ su ต้องใช้รหัสผ่านของบัญชีเป้าหมาย ในขณะที่ sudo ต้องใช้รหัสผ่านของผู้ใช้ปัจจุบัน ดังนั้นจึงปลอดภัยกว่าที่จะใช้ sudo เนื่องจากไม่มีการแลกเปลี่ยนข้อมูลที่ละเอียดอ่อน

นอกจากนี้ ขอแนะนำให้ใช้ sudo เมื่อทำงานที่ต้องใช้สิทธิ์รูท ผู้ใช้ปัจจุบันจะได้รับสิทธิพิเศษสำหรับคำสั่งที่ระบุเท่านั้น ในทางกลับกัน su เปลี่ยนไปใช้ผู้ใช้รูทโดยสมบูรณ์ ทำให้ระบบทั้งหมดถูกแก้ไขโดยไม่ได้ตั้งใจ

**How to Use the su Command**

su [user\_name] หรือ su - [user\_name] หากใช้คำสั่งโดยไม่มีอาร์กิวเมนต์ คำสั่งจะสลับไปยังบัญชี superuser (root)



คำสั่ง su ทำงานอย่างไร

หากต้องการเรียกใช้เชลล์ของผู้ใช้รายอื่นภายในไดเร็กทอรีการทำงาน/สภาพแวดล้อมผู้ใช้ ให้ใช้คำสั่ง su (ไม่มียัติภังค์)

ตัวอย่างเช่น หากต้องการใช้งานในฐานะผู้ใช้ชื่อ phoenixnap ให้รัน:

su phoenixnap

จากนั้นระบุรหัสผ่านสำหรับบัญชี phoenixnap และกด Enter

เชลล์ควรเปลี่ยน โดยแสดงว่าตอนนี้คุณมีสิทธิ์เข้าถึงบัญชีที่ระบุโดยโอเปอเรเตอร์ อย่างไรก็ตาม สภาพแวดล้อมของผู้ใช้ยังคงเหมือนเดิม ดังภาพด้านล่าง:

How to use the su command.

How to Use the sudo Command

sudo ใช้เป็นคำนำหน้าคำสั่ง Linux ซึ่งอนุญาตให้ผู้ใช้ที่เข้าสู่ระบบดำเนินการคำสั่งที่ต้องการสิทธิ์รูท ซึ่งแตกต่างจาก su คำสั่ง sudo ใน Linux ต้องระบุรหัสผ่านสำหรับผู้ใช้ที่เรียกใช้คำสั่ง

งานด้านการดูแลระบบและปฏิบัติการทั้งหมดต้องได้รับการอนุญาตสูงสุด (ถือโดยรูท) ในกรณีเช่นนี้ ขอแนะนำให้ใช้ sudo

ไวยากรณ์หลักคือ: sudo [command]

!!!อ๊ะๆๆ แต่เดี๋ยวก่อนนน ก่อนที่ระบบจะดำเนินการคำสั่ง ระบบจะถามรหัสผ่านของผู้ใช้ปัจจุบัน

โปรดทราบว่าตัวเลือก sudo สามารถใช้ได้โดยผู้ใช้ที่อยู่ในกลุ่ม sudoers เท่านั้น

**เรามาดูขั้นตอนการเพิ่มผู้ใช้ในกลุ่ม Sudoers กัน!!**

เพื่อให้ผู้ใช้เรียกใช้คำสั่งที่ต้องใช้คำนำหน้า sudo คำสั่งนั้นจะต้องเป็นส่วนหนึ่งของกลุ่ม sudoers หากต้องการเพิ่มผู้ใช้ในกลุ่ม sudoers ให้รันคำสั่งต่อไปนี้ (ในฐานะรูทหรือบัญชีที่มีสิทธิ์ sudo อยู่แล้ว):

usermod -aG sudo [user\_name]

ตัวอย่างเช่น หากต้องการเพิ่มบัญชี phoenixnap คุณต้องพิมพ์:

sudo usermod -aG sudo phoenixnap

หากต้องการดูรายการบัญชีที่เป็นของกลุ่ม sudoers ให้เรียกใช้:

sudo getent group sudo

ผลลัพธ์ควรแสดงบัญชีที่เพิ่มในขั้นตอนก่อนหน้า:

List users in the sudoers group.

อ้าวแล้วหลังจากที่ได้รู้แล้วว่า Su กับ Sudo ใช้ยังไง แล้วมันแตกต่างกันยังไงนะ?

เรามาดูกันว่า sudo และ su ทำงานบน Linux ดิสทริบิวชันที่ต่างกันอย่างไร

su เป็นคำสั่งที่เก่ากว่าแต่มีคุณสมบัติครบถ้วนมากกว่าซึ่งรวมอยู่ในลีนุกซ์รุ่นดิสทริบิวชันทั้งหมด เป็นวิธีดั้งเดิมในการเปลี่ยนไปใช้บัญชีรูท

Linux ไม่สนับสนุนการทำงานในฐานะรูทเนื่องจากอาจทำให้เกิดการเปลี่ยนแปลงทั้งระบบที่ไม่ต้องการและแนะนำให้ใช้ sudo แทน ด้วยเหตุนี้ การเผยแพร่บน Ubuntu ทั้งหมดจึงเป็น sudo-only ซึ่งหมายความว่าบัญชีรูทจะไม่ทำงานตามค่าเริ่มต้น

ขณะติดตั้งระบบปฏิบัติการ Ubuntu คุณจะสร้างผู้ใช้ที่มีป้ายกำกับโดยอัตโนมัติว่าเป็นส่วนหนึ่งของกลุ่ม sudoers อย่างไรก็ตาม ไม่มีการตั้งค่าบัญชีรูท ในการเปิดใช้งานผู้ใช้รูท คุณต้องเปิดใช้งานด้วยตนเอง

ในทางกลับกัน ลีนุกซ์รุ่นอื่นๆ เช่น Fedora จะสร้างรูทและบัญชีผู้ใช้เมื่อทำการติดตั้ง

ปล..!! อย่าลืมใช้ sudo ทุกครั้งที่ทำได้เพื่อป้องกันความปลอดภัยที่อาจเกิดขึ้นและปัญหาทั้งระบบนะคร้าบบบบ