

TP process Apache

Identification des processus apache

```
vboxuser@vbox:~$ ps aux | grep apache2
root      902  0.0  0.5 270140 37168 ?          Ss   09:35  0:00 /usr/sbin/apache2 -k start
www-data  934  0.0  0.2 270692 16776 ?          SN   09:35  0:00 /usr/sbin/apache2 -k start
www-data  935  0.0  0.2 270692 16776 ?          S    09:35  0:00 /usr/sbin/apache2 -k start
www-data  936  0.0  0.2 270692 16776 ?          S    09:35  0:00 /usr/sbin/apache2 -k start
www-data  937  0.0  0.2 270692 16776 ?          S    09:35  0:00 /usr/sbin/apache2 -k start
www-data  938  0.0  0.2 270692 16776 ?          S    09:35  0:00 /usr/sbin/apache2 -k start
vboxuser  4086 0.0  0.0   6336  2200 pts/0        S+   10:52  0:00 grep apache2
vboxuser@vbox:~$ top
```

Observation des processus via top

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
551	root	20	0	25356	8016	6948	S	0.0	0.1	0:00.52	systemd-logind
552	root	20	0	394844	14944	10372	S	0.0	0.2	0:00.30	udisksd
558	avahi	20	0	8108	364	0	S	0.0	0.0	0:00.00	avahi-daemon
587	root	20	0	258888	24192	16780	S	0.0	0.4	0:00.48	NetworkManager
588	root	-2	0	0	0	0	S	0.0	0.0	0:00.00	psimon
596	root	20	0	16540	5864	5000	S	0.0	0.1	0:00.07	wpa_supplicant
609	root	20	0	317320	12140	10292	S	0.0	0.2	0:00.23	ModemManager
683	root	20	0	27060	9000	7316	S	0.0	0.1	0:00.07	cupsd
685	root	20	0	267140	35760	27112	S	0.0	0.6	0:00.69	php-fpm8.2
700	root	20	0	1352968	51616	30596	S	0.0	0.8	0:04.64	containerd
791	root	20	0	15440	8812	7544	S	0.0	0.1	0:00.02	sshd
860	mysql	20	0	1349764	233960	24672	S	0.0	3.8	0:02.09	mariadb
873	root	20	0	292940	2936	2520	S	0.0	0.0	0:01.16	VBoxService
886	root	20	0	176248	15296	11240	S	0.0	0.2	0:00.12	cups-browsed
896	root	20	0	0	0	0	I	0.0	0.0	0:00.95	kworker/0:3-cgroup_desti
902	root	20	0	270140	37168	26936	S	0.0	0.6	0:00.52	apache2
903	root	20	0	240392	11152	7900	S	0.0	0.2	0:00.12	gdm3
912	www-data	20	0	267676	14760	6092	S	0.0	0.2	0:00.00	php-fpm8.2
913	www-data	20	0	267676	14760	6092	S	0.0	0.2	0:00.00	php-fpm8.2
934	www-data	30	10	270692	16776	6504	S	0.0	0.3	0:00.00	apache2
935	www-data	20	0	270692	16776	6504	S	0.0	0.3	0:00.02	apache2
936	www-data	20	0	270692	16776	6504	S	0.0	0.3	0:00.01	apache2
937	www-data	20	0	270692	16776	6504	S	0.0	0.3	0:00.00	apache2
938	www-data	20	0	270692	16776	6504	S	0.0	0.3	0:00.00	apache2

Et via htop après installation



Afficher la priorité (PR) et le nice (NI) des processus Apache :

```
vboxuser@vbox:~$ ps -o pid,ppid,cmd,ni,pri -C apache2
  PID      PPID CMD                      NI PRI
  902        1 /usr/sbin/apache2 -k start    0   19
  934      902 /usr/sbin/apache2 -k start   10   9
  935      902 /usr/sbin/apache2 -k start    0   19
  936      902 /usr/sbin/apache2 -k start    0   19
  937      902 /usr/sbin/apache2 -k start    0   19
  938      902 /usr/sbin/apache2 -k start    0   19
```

Modification de la priorité du worker au pid 934

```
vboxuser@vbox:~$ sudo renice -n 11 -p 934
934 (process ID) old priority 10, new priority 11
vboxuser@vbox:~$ █
```

Nice(ni) définit la "gentillesse" du processus à laisser du CPU aux autres.

- Valeur entre -20 (priorité haute) et 19 (priorité basse).
- renice permet de modifier la priorité d'un processus en cours.

Lancer un script qui consomme du CPU pour observer la concurrence :

```
while true; do echo "charge CPU" > /dev/null; done
```

Main	I/O	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
		2656	vboxuser	20	0	535M	83224	62592	S	0.0	1.3	0:00.29	/usr/libexec/gsd-xsettings
		2665	vboxuser	20	0	535M	83224	62592	S	0.0	1.3	0:00.00	/usr/libexec/gsd-xsettings
		2666	vboxuser	20	0	535M	83224	62592	S	0.0	1.3	0:00.02	/usr/libexec/gsd-xsettings
		2667	vboxuser	20	0	535M	83224	62592	S	0.0	1.3	0:00.00	/usr/libexec/gsd-xsettings
		2704	vboxuser	20	0	331M	25508	18112	S	0.0	0.4	0:00.15	/usr/libexec/xdg-desktop-portal-gtk
		2706	vboxuser	20	0	331M	25508	18112	S	0.0	0.4	0:00.00	/usr/libexec/xdg-desktop-portal-gtk
		2708	vboxuser	20	0	331M	25508	18112	S	0.0	0.4	0:00.00	/usr/libexec/xdg-desktop-portal-gtk
		2709	vboxuser	20	0	331M	25508	18112	S	0.0	0.4	0:00.02	/usr/libexec/xdg-desktop-portal-gtk
		2711	vboxuser	20	0	185M	22812	17884	S	0.0	0.4	0:00.10	/usr/libexec/ibus-x11
		2713	vboxuser	20	0	185M	22812	17884	S	0.0	0.4	0:00.00	/usr/libexec/ibus-x11
		2714	vboxuser	20	0	185M	22812	17884	S	0.0	0.4	0:00.00	/usr/libexec/ibus-x11
		2718	vboxuser	20	0	18692	400	4	S	0.0	0.0	0:00.00	/usr/bin/VBoxClient --clipboard
		2720	vboxuser	20	0	218M	29736	1900	S	0.0	0.5	0:00.13	/usr/bin/VBoxClient --clipboard
		2722	vboxuser	20	0	218M	29736	1900	S	0.0	0.5	0:00.00	/usr/bin/VBoxClient --clipboard
		2724	vboxuser	20	0	218M	29736	1900	S	0.0	0.5	0:00.00	/usr/bin/VBoxClient --clipboard
		2728	vboxuser	20	0	218M	29736	1900	S	0.0	0.5	0:00.12	/usr/bin/VBoxClient --vmsvga-session
		2744	vboxuser	20	0	18692	404	4	S	0.0	0.0	0:00.00	/usr/bin/VBoxClient --vmsvga-session
		2745	vboxuser	20	0	147M	404	4	S	0.0	0.0	0:00.56	/usr/bin/VBoxClient --vmsvga-session
		2746	vboxuser	20	0	147M	404	4	S	0.0	0.0	0:00.00	/usr/bin/VBoxClient --vmsvga-session
		2748	vboxuser	20	0	147M	404	4	S	0.0	0.0	0:00.56	/usr/bin/VBoxClient --vmsvga-session
		2778	vboxuser	20	0	295M	13216	9784	S	0.0	0.2	0:00.05	/usr/libexec/glib-pacrunner
		2779	vboxuser	20	0	295M	13216	9784	S	0.0	0.2	0:00.00	/usr/libexec/glib-pacrunner
		2781	vboxuser	20	0	295M	13216	9784	S	0.0	0.2	0:00.00	/usr/libexec/gnome-terminal-server
		2929	vboxuser	20	0	618M	62500	43112	S	0.0	1.0	0:34.81	bash
		4478	vboxuser	20	0	8248	5096	3480	R	116.0	0.1	2:45.67	
-	-	Next	S-F3	Prev	Esc	Cancel	Search:	bash					

Apache se fait voler du temps CPU. Modifier les Nice pour changer le comportement.

```
vboxuser@vbox:~$ sudo renice -n 12 -p 4478
[sudo] password for vboxuser:
4478 (process ID) old priority 0, new priority 12
vboxuser@vbox:~$ sudo renice -n -20 -p 4478
4478 (process ID) old priority 12, new priority -20
vboxuser@vbox:~$ sudo renice -n 19 -p 4478
4478 (process ID) old priority -20, new priority 19
vboxuser@vbox:~$ sudo renice -n -20 -p 902
902 (process ID) old priority 0, new priority -20
vboxuser@vbox:~$ sudo renice -n -20 -p 934
934 (process ID) old priority 11, new priority -20
```

En changeant les priorisées Nice, en rendant mon script plus Nice et apache moins Nice(donc prioritaire) apache devrait être prioritaire sur les ressources du CPU mais dans mon exo apache n'a pas besoin de plus de ressources donc le process du script continu a prendre tous le CPU.

Tuer un processus Apache worker

sudo kill -9 934

après le kill du processus apache a recréé un nouveau worker

avant :

```
CGroup: /system.slice/apache2.service
└─902 /usr/sbin/apache2 -k start
   ├─934 /usr/sbin/apache2 -k start
   ├─935 /usr/sbin/apache2 -k start
   ├─936 /usr/sbin/apache2 -k start
   ├─937 /usr/sbin/apache2 -k start
   └─938 /usr/sbin/apache2 -k start
```

Après

```
CGroup: /system.slice/apache2.service
└─902 /usr/sbin/apache2 -k start
   ├─935 /usr/sbin/apache2 -k start
   ├─936 /usr/sbin/apache2 -k start
   ├─937 /usr/sbin/apache2 -k start
   ├─938 /usr/sbin/apache2 -k start
   └─4579 /usr/sbin/apache2 -k start
```

- ◆ Étape 4 : Cas concret — Automatisation du redémarrage d'Apache

scénario : Vous êtes admin système, Apache s'est arrêté, vous devez vérifier et relancer automatiquement.

4.1 - Simuler l'arrêt d'Apache :

```
sudo systemctl stop apache2
```

4.2 - Appliquer et exécuter le script de supervision

Cree un fichier de supervision

```
Nano /home/vboxuser/script/apache_watchdog.sh
```

```
#!/bin/bash
```

```
if ! pgrep apache2 > /dev/null; then
    echo "$(date): Apache down, restarting..." >> /var/log/apache_watchdog.log
    systemctl start apache2
else
    echo "$(date): Apache OK" >> /var/log/apache_watchdog.log
fi
```

Explication du script

```
#!/bin/bash
```

Spécifie que le script doit être exécuté avec Bash

```
if ! pgrep apache2 > /dev/null; then  
    → Vérifie si Apache (processus apache2) n'est pas actif (pgrep retourne rien → alors on entre dans le if).  
    echo "$(date): Apache down, restarting..." >> /var/log/apache_watchdog.log  
    → Journalise la date et cree un message d'erreur dans un fichier de log personnalisé.  
    systemctl start apache2  
    → Tente de redémarrer Apache.  
else  
    echo "$(date): Apache OK" >> /var/log/apache_watchdog.log  
fi
```

→ Si Apache est déjà actif, journalise que tout va bien.

Rendre le script executable

```
chmod +x /home/vboxuser/script/apache_watchdog.sh
```

Tester le script : (je suis dans le répertoire /script)

```
vboxuser@vbox:~/script$ sudo ./apache_watchdog.sh  
vboxuser@vbox:~/script$ cat /var/log/apache_watchdog.log  
Fri Jun 27 12:07:54 PM CEST 2025: Apache OK  
vboxuser@vbox:~/script$ █
```

dans certaines distributions (notamment Debian/Ubuntu), le processus apache2 parent reste présent même si les workers sont arrêtés. Résultat : pgrep trouve encore un apache2, donc le script croit qu'Apache est actif.

Remplacement de la première variable if ! pgrep apache2 > /dev/null; then par

```
if ! systemctl is-active --quiet apache2; then
```

```
vboxuser@vbox:~/script$ cat /var/log/apache_watchdog.log  
Fri Jun 27 12:07:54 PM CEST 2025: Apache OK  
Fri Jun 27 12:20:35 PM CEST 2025: Apache down, restarting...
```

Pourquoi ça marche mieux ?

Méthode	Précision	Fiable ?
pgrep apache2	Cherche juste un PID	✗ Non (peut être trompeur)
systemctl is-active apache2	Vérifie l'état réel du service	<input checked="" type="checkbox"/> Oui

Planifier une vérification toutes les 2 minutes avec cron :

sudo crontab -e

***/2 * * * * /home/vboxuser/script/apache_watchdog.sh**

Test arrêter apach resultat redemarre tout seul au bout de deux minutes

Test ok