M1101 TD8
Gestion des processus

L'objectif de ce TD est de voir quelques commandes permettant de gérer les processus sous un système d'exploitation Linux.

1. La commande PS

Les processus sont un des éléments essentiels des applications. En effet lorsque l'on exécute une commande le noyau Linux créé un processus (tâche qui exécute les instructions de la commande). Chaque processus est identifié par une valeur unique (PID) qu'il garde quel que soit son état (actif, prêt, en attente). Les processus sont créés par une instruction noyau appelée « fork » qui clone le processus appelant la nouvelle fonctionnalité. On dit que le nouveau processus est le fils de celui qui l'a créé. Ce nouveau processus exécute alors un autre commande à l'aide d'un commande noyau « exec ». La commande PS permet d'afficher des informations sur les processus en cours d'exécution. Voici un extrait du manuel de la commande ps :

```
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session OR by effective group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
g  OBSOLETE -- DO NOT USE             -u by effective user ID (supports names)
r  only running processes             U  processes for specified users
x  processes w/o controlling ttys     t  by tty
********** output format **********  ********** long options **********
-o,o user-defined  -f full            --Group --User --pid --cols --ppid
-j,j job control   s  signal          --group --user --sid --rows --info
-O,O preloaded -o  v  virtual memory  --cumulative --format --deselect
-l,l long          u  user-oriented   --sort --tty --forest --version
-F   extra full    X  registers       --heading --no-heading --context
                  ********* misc options *********
-V,V  show version      L  list format codes  f  ASCII art forest
-m,m,-L,-T,H  threads   S  children in sum     -y change -l format
-M,Z  security data     c  true command name  -c scheduling class
-w,w  wide output       n  numeric WCHAN,UID  -H process hierarchy
```

Voici par exemple le résultat de la commande ps auxf

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2  0.0  0.0      0     0 ?        S    23:06   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    23:06   0:00  \_ [migration/0]
root         4  0.0  0.0      0     0 ?        S    23:06   0:00  \_ [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    23:06   0:00  \_ [watchdog/0]
root         6  0.0  0.0      0     0 ?        S    23:06   0:00  \_ [events/0]
root         7  0.0  0.0      0     0 ?        S    23:06   0:00  \_ [cpuset]
…
root         1  0.0  0.1   8356   660 ?        Ss   23:06   0:00 init [2]
root       224  0.0  0.0  17028   312 ?        S<s  23:06   0:00 udevd --daemon
root       302  0.0  0.0  17024   216 ?        S<   23:06   0:00  \_ udevd --daemon
root       318  0.0  0.0  17024   216 ?        S<   23:06   0:00  \_ udevd --daemon
daemon     686  0.0  0.1   8096   444 ?        Ss   23:06   0:00 /sbin/portmap
statd      705  0.0  0.1  14384   620 ?        Ss   23:06   0:00 /sbin/rpc.statd
root       862  0.0  0.3  62676  1204 ?        Sl   23:06   0:00 /usr/sbin/rsyslogd -c4
```

```
…
root      976  0.0  0.6  74336  2508 ?        Sl   23:06   0:00 /usr/sbin/gdm3
root      982  0.0  0.8  99796  3184 ?        Sl   23:06   0:00  \_ /usr/lib/gdm3/gdm-simple-slave --display-id /org/gnome/DisplayMana
root      988  1.6  6.1 118100 23200 tty7     Ss+  23:06   0:39      \_ /usr/bin/Xorg :0 -br -verbose -audit 0 -novtswitch -auth /var/
root     1605  0.0  0.7 101464  2816 ?        Sl   23:06   0:00      \_ /usr/lib/gdm3/gdm-session-worker
test     1680  0.0  1.9 158752  7556 ?        Ssl  23:07   0:00        \_ x-session-manager
test     1740  0.0  0.1  11880   400 ?        Ss   23:07   0:00          \_ /usr/bin/ssh-agent /usr/bin/dbus-launch --exit-with-se
test     1754  0.0  1.7 156092  6472 ?        Ss   23:07   0:00          \_ /usr/bin/seahorse-agent --execute x-session-manager
test     1762  0.0  2.5 214276  9724 ?        S    23:07   0:00          \_ gnome-power-manager
test     1772  0.1  3.4 225820 13264 ?        S    23:07   0:02          \_ /usr/bin/metacity
test     1779  0.1  5.1 371132 19572 ?        S    23:07   0:03          \_ gnome-panel
test     1785  0.1  4.4 421636 17056 ?        S    23:07   0:02          \_ nautilus
test     1790  0.0  2.0 146224  7704 ?        S    23:07   0:00          \_ bluetooth-applet
test     1793  0.0  3.1 286420 11988 ?        S    23:07   0:00          \_ nm-applet --sm-disable
test     1797  0.0  1.7 139032  6692 ?        S    23:07   0:00          \_ kerneloops-applet
test     1798  0.0  2.8 215200 10652 ?        Sl   23:07   0:00          \_ /usr/lib/policykit-1-gnome/polkit-gnome-authentication
test     1799  0.0  2.9 219744 11392 ?        S    23:07   0:00          \_ update-notifier
test     1801  0.0  2.3 264344  8820 ?        S    23:07   0:00          \_ /usr/lib/evolution/2.30/evolution-alarm-notify
test     1802  0.0  1.9 167624  7304 ?        S    23:07   0:00          \_ /usr/lib/gnome-disk-utility/gdu-notification-daemon
test     1804  0.0  4.9 222464 18624 ?        S    23:07   0:00          \_ python /usr/bin/system-config-printer-applet
…
root     1876  0.0  0.1   6756   680 ?        Ss   23:08   0:00 dhclient -v -pf /var/run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.
root     2034  0.0  0.2   6756   820 ?        Ss   23:25   0:00 dhclient eth0
test     2065  5.5 24.9 531276 94884 ?        Sl   23:27   1:06 /usr/lib/iceweasel/firefox-bin
test     2196  1.0  3.8 232684 14692 ?        Sl   23:28   0:11 gnome-terminal
test     2197  0.0  0.2  14332   780 ?        S    23:28   0:00  \_ gnome-pty-helper
test     2198  0.0  1.0  21084  3888 pts/0    Ss   23:28   0:00  \_ bash
root     2212  0.0  0.4  55856  1764 pts/0    S    23:28   0:00  |   \_ su
root     2220  0.0  0.5  19312  2108 pts/0    S    23:28   0:00  |     \_ bash
root     2226  0.0  0.4  18752  1784 pts/0    S+   23:29   0:00  |       \_ man ps
root     2238  0.0  0.2   9880  1032 pts/0    S+   23:29   0:00  |         \_ pager -s
test     2259  0.0  1.0  21084  3896 pts/1    Ss   23:30   0:00  \_ bash
test     2326  0.0  0.3  16452  1168 pts/1    R+   23:47   0:00      \_ ps auxf
```

1. Quel est le PID du process init ?
2. Quel est le processus père du processus gdm3 ?
3. Combien de terminaux sont ouverts dans la session graphique ?
4. Quel est le pid de la commande ps auxf lancée dans un des terminaux ?
5. Quel est l'utilisateur qui a lancé le man de la commande ps ?
6. Quel est le mode d'attribution de l'adresse pour l'interface eth0 ?
7. Que se passe-t-il pour le processus 2326 si on détruit le processus portant le pid 2259 ?

Les commandes & et kill

La commande & permet de lancer une application en tache de fond. Elle n'est plus bloquante vis-à-vis du terminal qui l'a exécutée. La commande kill quant à elle permet d'envoyer un signal à un processus. Généralement on l'utilise avec l'option -15 pour terminer un processus proprement, -9 pour terminer un processus (solution brutale qui ne laisse aucune chance au processus d'effectuer une quelconque action). Voici la liste des processus lancés. Wireshark a été lancé normalement en tâche principale.

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
…
test      2556  0.5  3.6 231840 13880 ?        Sl   00:27   0:01 gnome-terminal
test      2557  0.0  0.2  14332   780 ?        S    00:27   0:00  \_ gnome-pty-helper
test      2558  0.0  1.0  21244  3968 pts/1    Ss   00:27   0:00  \_ bash
root      2579  0.0  0.4  55856  1760 pts/1    S    00:29   0:00      \_ su
root      2588  0.0  0.5  19312  2092 pts/1    S    00:29   0:00        \_ bash
root      2637  0.0  0.3  16452  1172 pts/1    R+   00:32   0:00          \_ ps auxf
test      2603  0.2  1.9 167176  7340 ?        SL   00:32   0:00 gksu /usr/bin/x-terminal-emulator
root      2608  0.1  0.4  55856  1764 pts/0    Ss+  00:32   0:00  \_ /bin/su root -c /usr/lib/libgksu/gksu-run-helper "/usr/bin/x-terminal-e
root      2617  0.0  0.1   9024   728 pts/0    S+   00:32   0:00    \_ /usr/lib/libgksu/gksu-run-helper /usr/bin/x-terminal-emulator
root      2621  0.0  0.1   3956   580 pts/0    S+   00:32   0:00      \_ sh -c /usr/bin/x-terminal-emulator
```

```
root     2622  1.5  3.4 224984 13208 pts/0    Sl+  00:32   0:00          \_ gnome-terminal
root     2630  0.0  0.2  14332   780 pts/0    S+   00:32   0:00          \_ gnome-pty-helper
root     2631  0.0  0.5  19312  2072 pts/2    Ss   00:32   0:00          \_ bash
root     2635  7.2 17.4 350452 66452 pts/2    S+   00:32   0:00              \_ wireshark
```

1. Que se passe-t-il pour le processus 2635 si on exécute la commande kill -9 2631 ?
2. Que se passe-t-il pour le processus 2631 si on exécute la commance kill -9 2635 ?
3. Si on exécute wireshark en tâche de fond (wireshark &) on obtient la liste des tâches suivante :

```
test     2603  0.0  1.9 167176  7340 ?        SL   00:32   0:00 gksu /usr/bin/x-terminal-emulator
root     2608  0.0  0.4  55856  1764 pts/0    Ss+  00:32   0:00  \_ /bin/su root -c /usr/lib/libgksu/gksu-run-helper "/usr/bin/x-terminal-e
root     2617  0.0  0.1   9024   728 pts/0    S+   00:32   0:00      \_ /usr/lib/libgksu/gksu-run-helper /usr/bin/x-terminal-emulator
root     2621  0.0  0.1   3956   580 pts/0    S+   00:32   0:00          \_ sh -c /usr/bin/x-terminal-emulator
root     2622  0.0  3.4 224984 13220 pts/0    Sl+  00:32   0:00              \_ gnome-terminal
root     2630  0.0  0.2  14332   780 pts/0    S+   00:32   0:00              \_ gnome-pty-helper
root     2631  0.0  0.5  19312  2108 pts/2    Ss+  00:32   0:00              \_ bash
root     2642 11.6 17.4 350388 66452 pts/2    S    00:38   0:00                  \_ wireshark
```

4. Si on tue le processus 2631 on note les informations suivantes

```
root@DebianFred_serv:/home/test# ps -aef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Sep22 ?        00:00:00 init [2]
root         2     0  0 Sep22 ?        00:00:00 [kthreadd]
root         3     2  0 Sep22 ?        00:00:00 [migration/0]
root         4     2  0 Sep22 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 Sep22 ?        00:00:00 [watchdog/0]
root         6     2  0 Sep22 ?        00:00:00 [events/0]
...
root      2642     1  0 00:38 ?        00:00:00 wireshark
root      2652  2588  0 00:42 pts/1    00:00:00 ps -aef
```

A quoi correspond le PID du processus père du processus wireshark ?

La commande CRON

La commande crontab permet de lancer à intervalles réguliers des commandes. La liste des commandes est stockée dans un fichier cron table. Ce dernier contient 6 champs Le 5 premiers définissent la périodicité d'exécution de la commande le sixième la commande.

Minutes	Heures	Jours du mois	Mois	Jours de la semaine	Commande

On peut utiliser un liste de jours en séparant les jours souhaités par des virgules (1, 3, 5). On peut utiliser un intervalle avec un – (1-3 pour les jours de la semaines représente du lundi au mercredi). Le caractère * correspond au plus grand intervalle (* correspond à 0-6 dans les jours de la semaine). Pour appliquer les requêtes d'un fichier cron il suffit, sous réserve d'avoir les bons droits, de lancer

crontab MonFichierCron

1. Expliquer ce que font les lignes de commande suivante dans le fichier MonFichierCron

*	*	*	*	*	date >> /tmp/log 2&1
0	21	*	*	*	who
0	0	1	*	1-6	cal
30	12	3,8,15	3-10	*	touch .bash_profile

2. Quelle ligne devriez vous rajouter pour lancer tous les jours à 0h1m un reset des adresse ip de l'interface eth0 ?
3. Quelle ligne devriez vous rajouter pour effectuer le 15 de chaque mois à 23h59 une copie de tous les répertoires user (/home) vers un disque externe (/dev/sdb1) ?