TP Linux: Planification de tâches

Consignes :

- Ouvrez ce document dans le lecteur pdf installé sur la machine (PAS dans le navigateur).
- Répondez aux questions en tapant votre réponse dans les champs du document.
- Sauvegarder le fichier en `TP-cron.pdf`.
- Déposer le fichier sur l'espace de cours dédié, sur le cours "R504-R509".

1 Introduction

Sur un serveur, certaines tâches doivent être exécutées de façon régulières (backup, test d'intégrité, mises à jour, etc.). On peut aussi dans certains cas souhaiter exécuter une commande ultérieurement. Linux dispose d'outils permettant ceci.

2 Executer une commande ultérieurement

La commande `at` permet de lancer une commande soit à un instant précis, soit dans un laps de temps donné.

Q2.1 - Préalable : pour que cette commande fonctionne, il faut que le paquet `at` soit installé, et que le service associé (`atd`) soit actif.
Sur une Debian récente, la commande pour gérer les services est `systemctl`. Donnez la ligne utilisant cette commande associée au filtre grep et permettant de vérifier l'état du *daemon* `atd` :

Q2.1 systemctl status atd

Vous devriez obtenir :

```
atd.service loaded active running Deferred execution scheduler
```

Si vous n'avez rien, c'est que le service n'est pas installé. Installez-le via le paquet `at`.
Il est possible qu'il faille activer le service, ceci se fait avec :

```
$ sudo systemctl start atd
```

Vérifier son état avec : `$ sudo systemctl status atd`

Q2.2 - Vérifier la date et l'heure avec `$ date`

Q2.3 - On souhaite programmer la création d'un fichier (vide) nommé AAAA dans 2mn. Quelle est la commande à exécuter pour créer un fichier ? :

Q2.3 at now + 2 minutes puis touch A

Q2.4 - Pour programmer cette commande, on tape `at HH:MN` (avec HH l'heure actuelle et MN la valeurs des mn. actuelle + 2). La commande vous propose une saisie (prompt `>`) : tapez la commande précédente, puis CTRL-D (code ASCII "EOT", *End Of Transmission*).

Q2.5 - Attendez et vérifiez que la commande a été exécutée.

Q2.6 - On peut aussi spécifier le délai via des mots clés. Par exemple, lancer les 4 jobs suivants :

```
at now +10 minutes
at now +10 hours
at now +10 days
at now +2 weeks
```

Pour chaque job, vous donnerez comme tâche `echo "coucou 10mn">t_XXX`,
avec "XXX" remplacé par `10mn`, `10h`, `10j`, `sem`
Note : on demande comme tâche la création d'un fichier parce que par défaut la commande n'est pas exécutée dans la console de l'utilisateur courant. On ne peut donc pas avoir un affichage à l'écran via "echo".

Q2.7 - Visualisez la liste des jobs avec la commande `atq`. Retrouvez-vous la liste précédente ?

Q2.8 - La commande `atrm` permet de supprimer un job programmé via `at`.
Donnez la commande permettant de supprimer le job prévu pour dans 2 semaines :

Q2.8 atrm <num>

Q2.9 - Quitter la console et la réouvrir. Les jobs sont-ils toujours présents dans la file d'attente ?

Q2.9 oui

Q2.10 - On peut visualiser la commande qui sera effectivement lancée avec `at -c <num>`, avec `<num>` le numéro du job, obtenu via `atq`. Afficher la commande prévue dans 10 jours.
Relancer cette commande en la passant dans le filtre `tail`.
A quoi correspondent les lignes avant la commande désirée ?

Q2.10 at -c 6 | tail

Q2.10b Cela définit des variable d'environnement en Bash

Q2.11 - Il est aussi possible d'ajouter des jobs de façon non-interactive, en précisant le nom du script qui devra être exécuté. Créer un script nommé `s1.sh` contenant juste la ligne `echo "coucou">test`.
(Note : il n'est pas nécessaire de rendre ce script exécutable, vu qu'il sera exécuté par un interpréteur.)
Tapez la commande `at now +1 minutes -f s1.sh`, et vérifiez son bon fonctionnement.

Q2.12 - On peut faire un sorte d'avoir un affichage dans la console en spécifiant le "device" de la console courante. On obtient cette information avec la commande `tty`. On peut ensuite avoir un affichage dans la console en redirigeant la sortie de "echo" vers ce périphérique.
Faire l'essai en demandant l'affichage de "Bonjour" dans 1 minute.

3 Programmation de tâches périodiques

L'outil `cron` permet de programmer de façon plus générale des tâches à exécuter à une échéance donnée, et de façon répétitive. Son utilisation est plus complexe que `at` : la programmation des jobs se fait via l'édition d'un fichier de configuration auquel on accède via la commande `crontab`.

Dans ce fichier, chaque job est décrit par une ligne. Chaque ligne du fichier contient 6 champs séparés par SPC (ASCII 32). Les 5 premiers déterminent les moments d'exécution de la tâche décrite au 6ème champ.

```
# ┌──────────── min (0 - 59)
# │ ┌────────── hour (0 - 23)
# │ │ ┌──────── day of month (1 - 31)
# │ │ │ ┌────── month (1 - 12)
# │ │ │ │ ┌──── day of week (0 - 6) (0 to 6 are Sunday to
# │ │ │ │ │     Saturday, or use names; 7 is also Sunday)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *  command to execute
```

Un champ temporel peut contenir :

- une valeur précise et valide pour le champ (par exemple 15 sur le champ minute)
- une liste de valeurs valides, séparées par des virgules (1,3,5 dans le champ mois : janvier, mars, mai)
- un intervalle valide (1-5 dans le champ jour : du lundi au vendredi)
- `*` pour signifier toutes les valeurs possibles du champ (* dans le champ minute : toutes les minutes)
- `*/5` (dans le champ minutes : tous les 5 minutes)

Pour installer l'exécution périodique de tâches relatives à la gestion du système sans contrainte particulière, il existe les répertoires suivants :

- `/etc/cron.hourly` pour une exécution toutes les heures ;

— `/etc/cron.daily` pour une exécution tous les jours ;
— `/etc/cron.weekly` pour une exécution hebdomadaire ;
— `/etc/cron.monthly` pour une exécution mensuelle ;

Leur utilisation est simple : il suffit d'y copier un fichier exécutable (ou de créer un lien) afin d'en activer l'exécution périodique.

Attention :

— les tâches s'éxecutent sans console attachée (stdin/stdout), on ne pourra donc pas avoir d'affichage.
— cron ne prévoit pas de système d'alerte si le lancement de la tâche échoue (erreur de syntaxe, machine éteinte, etc.)

Q3.1 - Préalable : pour que cette commande fonctionne, il faut que le service associé (`cron`) soit actif. Donnez la commande utilisant `systemctl` permettant de vérifier l'état du *daemon* :

Q3.1 systemctl status cron

Q3.2 - Donner une autre façon, en vérifiant via `ps aux` si le *daemon* `crond` est actif :

Q3.2 ps aux | grep crond

Q3.3 - En utilisant `man`, notez dans le tableau ci-dessous les principales options à utiliser avec la commande `crontab`.

Q3.3a crontab -l : *display current crontab file*

Q3.3b crontab -r : *erase current crontab file*

Q3.3c crontab -e : *edit current crontab file*

Q3.4 - Créer une tâche qui devra toutes les minutes ajouter une ligne dans un fichier `~/cron-date.txt` (placé dans votre "home") contenant la date et l'heure. Donner la ligne à ajouter au fichier cron :

Q3.4 */1 * * * * echo "coucou2">>~/cron-date.txt

Q3.5 - Le fichier crontab principal est `/etc/crontab`. Afficher ce fichier avec `cat`. Retrouvez-vous votre tâche planifiée ? Q3.5 Non

Q3.6 - Le fichier précédent est réservé pour les tâches système. Les tâches utilisateurs sont (sur Debian) localisées dans `/var/spool/cron/crontabs/`. Donner la commande permettant de vérifier la présence de votre fichier :

Q3.6 ls /var/spool/cron/crontabs/

Q3.7 - Donner la commande permettant d'afficher ce fichier :

Q3.7 cat /var/spool/cron/crontabs/fi25-pinel

Q3.8 - Un administrateur peut autoriser/interdire à certains utilisateurs de planifier des tâches . Il faut pour cela placer ces identifiants dans les fichiers `/etc/cron.allow` et/ou `/etc/cron.deny`.
Créer un utilisateur `user3` (avec la commande `adduser`) et placer son nom dans `/etc/cron.deny`, puis vérifier qu'il ne peut pas créer de fichier crontab.

4 Anacron

Le *daemon* `cron` est parfait pour un serveur, qui tourne 24/24. Mais dans d'autres contextes (*workstation*), il a le défaut de considérer que la machine est toujours allumée : une tâche planifiée à un moment où la machine est éteinte sera ignorée. De plus, aucune trace ne sera laissée de l'absence d'exécution de cette tâche.

Pour pallier ce défaut, un autre utilitaire doit être utilisé [1] : `anacron`

Ce programme va lancer les tâches programmées qui n'ont pu être effectuées dès que la machine redémarre. Par contre, il n'autorise pas une gestion temporelle aussi fine que `cron`, il se limite au tâches journalières ou plus. Et il ne peut être programmé que par le user "root".

1. qui techniquement n'est pas implémenté comme un *daemon* ; voir `https://fr.wikipedia.org/wiki/Anacron`

Q4.1 - Visualisez le fichier principal `/etc/crontab`. Vous y trouvez 4 tâches à lancer. Analyser la commande et donner la périodicité de chacune d'entre elles dans le tableau ci-dessous :

Ligne	Périodicité
1	Q4.1 toutes les 17 min
2	Q4.1 toutes les 17 min
3	Q4.1 toutes les 17 min
4	Q4.1 toutes les 17 min

Q4.2 - Expliquez la commande qui est lancée pour les deux premières. Il faudra surement vous aider de `man` pour documenter les commandes que vous ne connaissez pas.

Q4.2a la premiere commande se déplace et lance un script dans /etc/cron.hourly

Q4.2b la deuxième commande test regarde le type des fichiers dans /usr/sbin/anacron avec le -x ce qui ont déj