Tâches à temps

—Rapport de projet de SE

Classe:1000

Membre du Groupe:

Nom/Prénom	**Numéro d'étudiant**
王泓哲(WANG Hongzhe)	18124533
胡瑞祺(HU Ruiqi)	18124565

Sommaire

1、Introduction du projet

Ce projet nous permet d'imiter la fonction de cront pour concevoir un script, implémenter deux commandes tacheron et tacherontab, tacheron est un processus démon utilisé pour effectuer périodiquement une certaine tâche ou attendre de traiter certains événements sous linux, le processus tacheron sera vérifié régulièrement S'il y a une tâche à exécuter, s'il y a une tâche à exécuter, la tâche sera exécutée automatiquement. Le contrôle tacheron est basé sur le fichier tacherontab de chaque utilisateur. Nous pouvons éditer ce fichier avec la commande tachrontab.

2、Division et plan du travail

Wanghongzhe(50%):
Principalement responsable du tachéron
Réalisation de la fonction principale de effectuer des tâches sur une base régulière
et implémentation de chaque fonction de détection de temps.
Concevoir l’interaction utilisateur (par example détecter la présence de fichiers critiques et signaler les erreurs)

Huruiqi(50%):
Principalement responsable du tachérontab
La réalisation de la fonction tacherontab;
Ajouter du contenu pour contrôler l'autorisation d'exécution de l'utilisateur dans le script tacheron;
Déboguer deux commandes et optimiser la structure syntaxique du script;
Coopérer pour terminer le rapport de projet.

3、structure du code

3.1 Partie Tacheron

3.1.1 Fonction principale

La commande tacheron est chargée de exécuter toutes tâches (commandes et scripts) définies dans le répertoire /etc/tacheron/.

Il lit régulièrement les fichiers présents dans le répertoire /etc/tacheron/ et le pour voir si des tâches doivent être exécutées. La structure du donnée dans le fichier est “15 secondes (0-3), minutes (0-59), heures (0-23), jour du mois (1-31), mois de l'année (1-12), jour de la semaine (0-6, 0=dimanche) ”

Cette commande peut être exécutée en arrière-plan sans interruption(nohup) pour gérer les tâches récurrentes.

3.1.2 La principale méthode de mise en œuvre

On utilise une boucle morte pour ce faire lire régulièrement les fichiers présents dans le répertoire ./etc/tacheron/

pour chaque cycle, on lit l’heure définie dans le fichier et la comparons au temps maintenant , si ils sont même ,on fait le commande après ce temps. Et à la fin ,on pause 15 secondes pour réaliser l’intervalle de temps minimum.

On doit verifier ce qui exécute le programme,le root peut accomplir toutes les tâches dans /etc/tacheron,mais d'autre utilisateur ne peut accomplir que sa propre tâche dans son propre dossier,c’est afin de restreindre les autorisations d’éviter d’effectuer le même tâche plusieur fois.

3.1.3 Explications du code

A.La fonction “meme_ou_pas”

Pour déterminer si le format est correct, divisez-la en six situations et discutez-en

1. Si le format de temps de la tâche est '*', on retourne 1 directement.

```
Type0 = "*"
```

2.Si le format de temps de la tâche est un nombre entier compris entre 0 et 99,on compare si le deux temps sont mêmes.

```
type1=$(echo $time1|grep '^[0-9][0-9]\?$')
```

3. Si le format de temps de la tâche est est un intervalle de temps connecteé par '-',on obtient les temps maximum et temps minimum,ensuite détermine si le temps

maintenant est dans ce intervalle.

type2=$(echo $time1|grep '^[0-9][0-9]\?-[0-9][0-9]\?$')

4. Si le format de temps de la tâche est est un série d'entiers séparés par ',',on lit toutes les possibilités,tant que l'une de ces conditions est adaptable, retourne 1.

type3=$(echo $time1|grep '^[0-9][0-9]\?\(,[0-9][0-9]\?\)*$')

5. Si le format de temps de la tâche est dans une formation comme '1-12~4~5',Sur la base de la situation 3,on verifie si le temps n'est pas même avec touts les numeros après '~'.

type4=$(echo $time1|grep '^[0-9][0-9]\?-[0-9][0-9]\?\(~[0-9][0-9]\?\)*$')

6. Si le format de temps de la tâche est dans une formation comme '0-23/3',nous calculons l'intervalle entre l'heure actuelle et le moment où le programme commence à s'exécute, si il est une multiplication du nombre après slash,et il est entre l'intervalle avant de slash aussi ,On retourne 1.

type5=$(echo $time1|grep '^.*-.*/.*$')

7.Si le format de temps de la tâche est dans une formation comme '*/number'，nous calculons l'intervalle entre l'heure actuelle et le moment où le programme commence à s'exécute, si il est une multiplication du nombre après slash. On retourne 1.

type6=$(echo $time1|grep '^.*/.*$')

Une fois qu'il est constaté que valeur d'un élémentde type n'est pas vide, il le compare avec le temps maintenant,Si ils sont mêmes,on retourne 1,sinon, retourne 0.

Et chaque fois, la comparaison doit être comparée six fois, si chaque valeur de retour est de 1,on le fait.

```
j=1;flagg=0
while [ $j -le 6 ]
      do
      exe_ou_pas (parametres)
      f [    $? -eq 1    ]
         flagg=$((flagg+1))
         fi
         j=$(( j + 1 ))
done
 if [ $flagg -eq 6 ]
 then
      ...(faire la tache)
```

fi

B. Le corps de fonction principale

Pour administateur,nous utilisons deux instructions en boucle while pour obtenir une lecture répétée des fichiers, et pour d'autre utilisateur ,un while est suffisant.
nous utilisions la commande date pour obtenir le temps et la commande pour obtenir le utilisateur maintenant.

```
time_start=$(date+"%S%M%H%d%m%w"|awk'{printf("%d",$1/15);print$2,$3,$4,$5,$6}')
dir_list=$(ls /etc/tacheron)

        while true                              done
        do                                    else
        user=$(whoami)                          while read t
        if [ $user = "root" ]                   do
        Then                                        ...
            ...                                 done < $list
        for d in $dir_list                    fi
          do                                sleep 15
              while read t                    done
              do
                 ....
```

C. Le format de sortie

```
if [ $flagg -eq 6 ]
then
   echo "L'heure actuelle est $time"
   echo -n "\""
   echo "$t"|awk 'BEGIN{} {for(i=7;i<=NF;i++) printf $i" "} END{}'
   mission=$(echo "$t"|awk '{for(i=7;i<=NF;i++) print $i" "}')
   echo "\" is done "
   echo "Les résultats de la mise en œuvre sont les suivants:"
   $mission
   echo "-------------------------------------------------------------------"
   echo "time:$time    " >> /var/log/tacheron
   echo "$t"|awk 'BEGIN{} {for(i=7;i<=NF;i++) printf $i" "} END{}' >> /var/log/tacheron
fi
```

La sortie est indiquée dans la figure ci-dessous

L'heure actuelle et les tâches exécutées sont la sortie d'abord,ensuite, nous allons montrer les résultats de l'exécution à l'écran.

Enfin, nous redirigions le temps et les tâches vers le fichier /var/log/tacheron.

```
[root@localhost Desktop]# ./tacheron.sh
L'heure actuelle est 3 09 12 07 03 0
"echo 1 " is done
Les résultats de la mise en œuvre sont les suivants:
1
------------------------------------------------------------
```

3.2 Partie Tacherontab

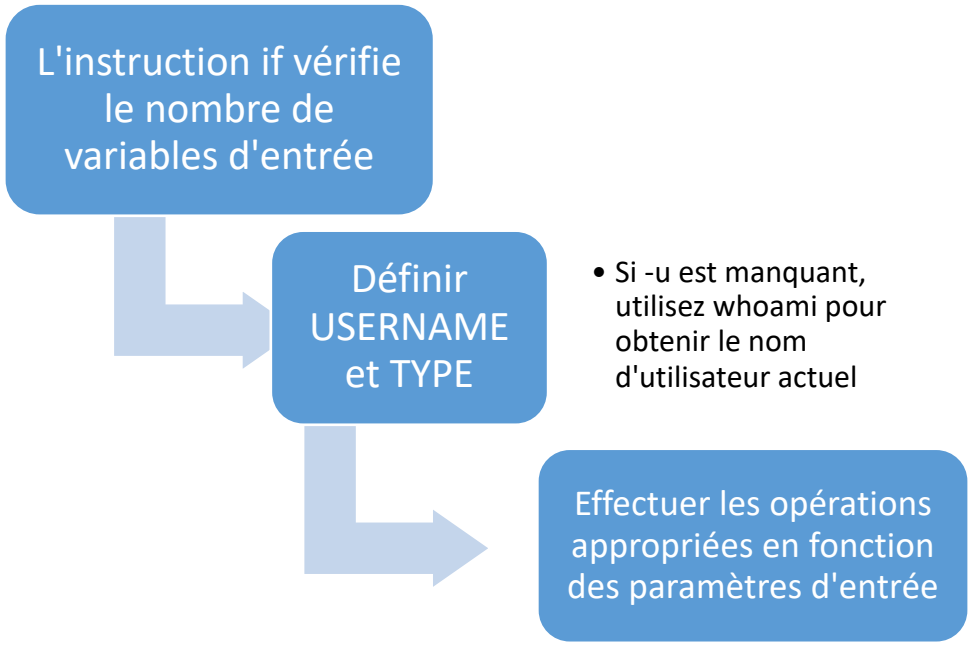
3.2.1 Fonction principale

Le but de la commande tacherontab est de modifier, lire et supprimer le fichier tacherontab pour chaque utilisateur. Il a quatre options, l'option optionnelle -u est utilisée pour spécifier le nom d'utilisateur, s'il manque, la valeur par défaut est l'utilisateur courant, l'option -l est utilisée pour lire le contenu de la table utilisateur, l'option -r est permet de supprimer la table utilisateur, - L'option e permet d'éditer et de modifier la table utilisateur.
Il est à noter que nous utilisons l'éditeur vi pour modifier les fichiers temporaires dans / tmp. Une fois les fichiers temporaires enregistrés, les fichiers temporaires sont copiés dans le répertoire / etc / tacheron.

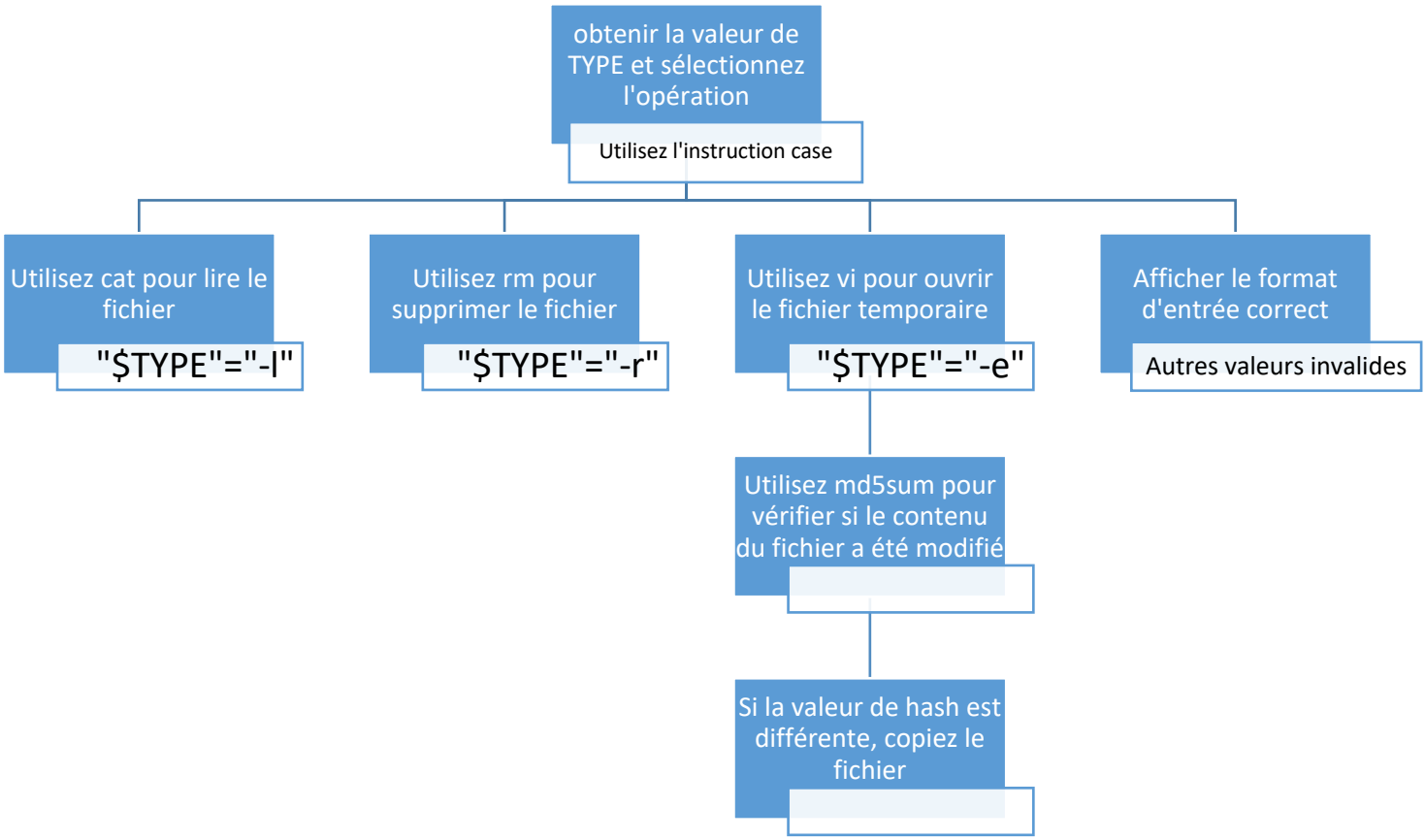
3.2.2 La principale méthode de mise en œuvre

Le script est principalement divisé en deux parties. La première consiste à juger des options et paramètres d'entrée. Lors de l'exécution, il est principalement déterminé par le nombre de variables d'entrée si -u est omis, puis les paramètres entrants $ 1 ($ 2, $ 3) sont défini sur TYPE et USERNAME.
La seconde consiste à juger le TYPE et à effectuer l'opération correspondante, et à signaler une erreur lorsque le paramètre d'entrée est illégal, en demandant le format d'entrée correct.
Nous avons dessiné un organigramme pour rendre la structure du script plus Claire.

première partie



deuxieme partie



3.1.3 Explications du code

```
#!/bin/bash
# commande tacherontab
if [ $# -eq 3 -a "$1" = "-u" ]
then
USERNAME=$2
TYPE=$3
elif [ $# -eq 1 ]
then
TYPE=$1
USERNAME=`whoami`
else
echo "error pls check your nomber of arguments #tacherontab [-u
    user] {-l | -r | -e}"
exit 0
fi
```

La première partie utilise principalement l'instruction if, utilise $ # pour déterminer le nombre de paramètres de script entrants et définit les paramètres du numéro de série sur les variables USERNAME et TYPE. Notez que lorsque -u est manquant, nous utilisons la commande whoami pour obtenir le nom de l'exécuteur de script actuel, et lorsque le nombre de paramètres d'entrée est incorrect, nous affichons le format d'entrée correct pour rappeler à l'utilisateur et quitter le script.

La deuxième partie utilise principalement l'instruction case.Avant l'exécution, vérifiez si les répertoires requis / etc / tacheron et / etc / tacheron / tacherontab $ USERNAME existent. Et utilisez cat pour réaliser la fonction de navigation, rm pour réaliser la fonction de suppression et utilisez l'éditeur vi pour réaliser la fonction d'édition.

```
case $TYPE in
"-l") if [ ! -f /etc/tacheron/tacherontab$USERNAME ]
        then
        echo "/etc/tacheron/tacherontab"$USERNAME" pas existe"
        else
        cat /etc/tacheron/tacherontab$USERNAME
        fi;;
"-r")   sudo rm /etc/tacheron/tacherontab$USERNAME;;
"-e")   if [ ! -d /etc/tacheron ]
        then
        echo "/etc/tacheron pas existe"
            mkdir /etc/tacheron
        else
            touch /tmp/tmpfichier$USERNAME
            hash=$(md5sum /tmp/tmpfichier$USERNAME | cut -d' ' -f1)
            vi /tmp/tmpfichier$USERNAME
            newhash=$(md5sum /tmp/tmpfichier$USERNAME | cut -d' '
                -f1)
            if [ $hash != $newhash ]
                  then
                    sudo cp /tmp/tmpfichier$USERNAME
                        /etc/tacheron/tacherontab$USERNAME
            fi
        fi;;
*) echo "error pls check your notation #tacherontab [-u user] {-l |
    -r | -e}" exit 0 ;;
esac
```

La difficulté dans cette partie est de déterminer si le fichier temporaire dans / tmp est quitté en enregistrant. Je pense à la façon de vérifier si le fichier est cohérent avec le fichier de sauvegarde dans l'exercice TD5: utilisez la commande md5sum. Nous pouvons obtenir séparément la valeur HASH du fichier avant et après l'ouverture du fichier temporaire dans le répertoire / tmp avec vi. Si la valeur HASH avant et après ouverture est différente, cela signifie que le contenu du fichier temporaire a été modifié et sauvegardé . À ce stade, nous copions ensuite ce fichier dans / etc / tacheron / tacherontab $ USERNAME pour réaliser cette fonction.

4、Conclusion

Nous nous sommes d'abord inspirés de la conception de cron et av ons acquis une compréhension approfondie du principe de fonction nement de cron et des fonctions spécifiques qu'il peut accomplir. C ela nous donne un aperçu des résultats que le projet peut finaleme nt atteindre, et quelques idées pour la conception d'algorithmes. Ap rès cela, nous avons passé en revue les parties pertinentes du mat ériel et des TDs. Après notre discussion, nous avons finalement dé cidé de comparer l'heure actuelle avec l'heure prédéfinie pour déter miner si la commande doit être exécutée.

Ensuite, nous étions tous les deux responsables de la préparation d'une commande, et finalement intégrée et déboguée ensemble. Après de nombreuses modifications et ajustements, et l'ajout de fon ctions, la version finale du projet était terminée.

Grâce à la formation de ce projet, nous avons une compréhension plus profonde du fonctionnement du système Linux, et nous somm es plus à l'aise avec l'écriture de scripts. Avec l'accumulation de co nnaissances dans ce cours, je pense que nous serons plus compét ents dans la conception de programmes sur Linux à l'avenir. S'il y a une chance, nous améliorerons également le script tacheron cette fois, ajouterons plus d'invites et de vérifications et améliorerons l'ex périence utilisateur.