**ATELIER 10 – LES PROCESSUS.**

**LES PROCESSUS UNIX**

**Un processus est un objet dynamique qui correspond à l'exécution d'un programme.**

**Le chargement d'un programme est la phase durant laquelle celui-ci passe de l'état inerte à l'état actif.**

**Toutes les activités dans le système (demande explicite d'utilisateur, activités purement système visant à gérer les ressources) sont exécutées dans le contexte d'un processus.**

**1) On peut distinguer deux types de processus**:

**a-) les processus système** :

* **ne sont sous le contrôle d'aucun terminal**
* **ont comme propriétaire, l'utilisateur privilégié. Ils restent résidents en mémoire centrale en attente d'une requête.**
* **assurent un certain nombre de services généraux accessibles à tous les utilisateurs du système ou de systèmes distants en cas d'appartenance à un réseau. Ils remplissent des tâches qui ne sont pas intégrées dans le noyau.**
* **peuvent être créés soit au lancement du système (ou à son passage en mode multi-utilisateurs), soit à des dates fixées par l'administrateur du système.**

**Exemples :**

* **init : s'assure, entre autres tâches, que sur chaque ligne de communication série asynchrone à laquelle est connecté un terminal alphanumérique, existe un processus permettant la connexion des utilisateurs (processus getty) et recueille les processus orphelins (problème de l'élimination des processus zombis)**
* **cron : son rôle est de lancer à des dates spécifiées des commandes particulières (exemple : effectuer une copie de sécurité des informations tous les jours à 18 heures).**

**b ) les processus utilisateurs**.

* **dédiés à l'exécution d'une tâche particulière ayant normalement une durée de vie limitée**

**Si la plupart des processus des utilisateurs sont sous le contrôle d'un terminal, cela n'est absolument pas une caractéristique d'un processus utilisateur.**

**2) Caractéristiques d'un processus**.

**Quelques caractéristiques d'un processus sont:**

* **son identification** : **PID (Processus Identifier), un nombre entier. Cet identificateur est retourné par le noyau du système lors de la création du processus. Il est unique**.
* **l'identificateur du processus parent:** **PPID, un nombre entier**.
* **Ses propriétaires et ses groupes propriétaires** : **le propriétaire et le groupe effectifs peuvent être différents du propriétaire et du groupe réels qui représentent l'utilisateur ayant lancé l'exécution de disposer d'une autre identité afin de pouvoir accéder à des fichiers dont l'accès est limité (principes "set-user-id" et "set-gid-id").**
* **Eventuellement son terminal d'attachement** : **c'est-à-dire celui à partir duquel il a été démarré**.
* **sa priorité** : **indication sur le degré d'urgence du processus. Elle permet de déterminer l'ordre d'allocation du processeur lorsque plusieurs processus sont en attente d'exécution (cas de la multi-programmation).**
* **TIME** : **temps CPU en minutes et secondes**
* **S ou R: état du processus Suspend ou Run**

**Une autre caractéristique importante des processus est qu'ils peuvent s'exécuter dans deux modes différents** :

* **mode utilisateur (user mode)** : **dans ce mode le processus n'accède qu'à son espace d'adressage et exécute des instructions ordinaires appartenant au programme chargé (par exemple: affecter à un objet z la somme des valeurs des deux objets x et y).**
* **mode noyau ou système (kernel mode)** : **le processus exécute alors des instructions qui n'appartiennent pas au programme en cours mais au noyau.**

**Exemple : accès à des données externes à son espace d'adressage (essentiellement les tables du système).**

**Passage en mode noyau : par la demande de réalisation d'un appel système par le processus lui-même.**

**La lecture de données au clavier est effectuée par une primitive du système. L'exécution du processus passe au niveau noyau pour cette phase du traitement permettant au processus d'accéder à des zones particulières de la mémoire et de communiquer avec le gestionnaire de périphériques. Après avoir reçu les informations nécessaires, l'exécution du processus continue au niveau utilisateur.**

**3) Ordonnancement des processus**

**Lorsqu'un processus est lancé, le système doit gérer la mémoire et l'allocation du processeur lui étant accordée. Il fait appel à l'ordonnanceur (*scheduler* en anglais).Un système d'exploitation est préemptif lorsqu'il peut arrêter à tout moment n'importe quelle application pour passer à la suivante (exemple : Windows XP, Windows 7 et GNU/Linux sont des systèmes préemptifs). Il peut aussi être coopératif quand il permet à plusieurs applications de fonctionner et d'occuper la mémoire, et leur laissant le soin de gérer cette occupation (exemple : Windows 95, 98 et Millénium sont des systèmes coopératifs).**

**En résumé :**

* **multitâche préemptif** : le système d'exploitation garde le contrôle et se réserve le droit de fermer l'application.
* **multitâche coopératif** : le système d'exploitation laisse les applications gérer (cela concerne les systèmes d’exploitation anciens comme windows3.1)

Vous aurez donc compris que le multitâche coopératif est plus « dangereux » pour le système, car risque de blocage si une application fait des siennes.

Enfin, dernière chose à retenir : **les systèmes basés sur Unix sont des systèmes préemptifs**.

**4) Etats d'un processus et transition d'états**

**Un processus en cours d'exécution, exécute les instructions qui composent le programme et accède à ses propres données. Il dialogue avec le reste des ressources disponibles par l'intermédiaire des fonctions d'appel au système.**

* **un état transitoire** : **à sa création ou au cours de la création d'un processus fils**.
* **des états actifs** : **(actif noyau ou utilisateur correspondant à son mode d'exécution) : le processus détient la ressource UC, ses propres instructions s'exécutent.**
* **l'état prêt** (éligible) : **le processus est éligible par l'ordonnanceur (il dispose de toutes ses ressources)**
* **l'état en sommeil** : **le processus est en attente d'événement (horloge, montage d'un volume disque, résultat d'une autre tâche …). Dans cet état, le processus ne consomme pas de temps CPU. Il repasse à l'état prêt si l'événement se produit.**
* **l'état suspendu** : **mécanisme de job control spécifique à certains interpréteurs shell comme le ksh, csh. Ce mécanisme permet aux utilisateurs de ne voir que les processus qu'ils ont lancés sous le shell interactif dans lequel ils travaillent et qui constitue le cadre d'une session de travail. Il permet également d'arrêter et de relancer des processus, de les déplacer au premier plan ou en arrière plan, de les supprimer etc…**
* **l'état zombi** : **le processus est terminé mais son père n'a pas pris connaissance de sa terminaison. Le processus occupe un emplacement dans la table des processus (c'est la seule ressource qu'il utilise) et ne consomme aucun temps CPU.**

**En tapant la commande** **ps -afl, vous allez obtenir des informations concernant tous les processus** :

**– le propriétaire (UID) ;**

**– le numéro du processus (PID) ;**

**– le numéro du processus père (PPID) ;**

**– la priorité (PRI) ;**

**– la valeur de NICE (NI), ce qui contribue à sa priorité de programmation ;**

**– l’état (STAT) ;**

- **l’heure du début du processus en heures, minutes et secondes (STIME) ;**

**– le nom du device vers lequel il est dirigé (TTY) ;**

**– le temps passé dans la cpu (TIME) ;**

**– le nom de la commande exécutée (COMMAND).**

**L’état d’un processus peut être les suivants** :

**– D : le processus est i interruptible ;**

**– R : le processus est en train de s’exécuter ;**

**– S : le processus est endormi ;**

**– T : le processus est stoppé;**

**– Z : le processus est zombi : il est mort mais son père ne le sait pas.**

**D’autres informations peuvent aussi apparaître :**

**– W : le processus a été swappé ;**

**– < : le processus possède une priorité haute ;**

**– N : le processus possède une priorité basse ;**

**– L : le processus possède des pages mémoires verrouillées.**

**5) Exécution d'un processus**.

**Le shell est l'environnement utilisateur en mode texte sous Linux. C'est le programme qui se charge de lire et d'exécuter les commandes que l'utilisateur saisit. Classiquement, le shell est utilisé de manière interactive, c'est-à-dire que l'utilisateur dialogue avec le système par l'intermédiaire du shell. Il saisit les commandes, et le shell les exécute et affiche les résultats. Le shell le plus couramment utilisé sous Linux est sans aucun doute bash. En tout cas, c'est le shell par défaut que la plupart des distributions utilisent. Il est donc conseillé de connaître un petit peu ce que ce shell est capable de réaliser, et comment. Le shell bash est une évolution du shell sh, utilisé par quasiment tous les systèmes Unix. Son nom provient de l'abréviation de l'anglais « Bourne Again SHell », ce qui signifie qu'il s'agit effectivement d'une nouvelle variante du shell sh.**

**Au temps des interfaces graphiques complexes et sophistiquées, il peut paraître archaïque de vouloir encore utiliser des lignes de commandes pour utiliser un ordinateur. C'est en partie vrai, mais il faut savoir que les shells Unix sont extrêmement puissants et que les interfaces graphiques ne permettent toujours pas, même à l'heure actuelle, de réaliser toutes les tâches faisables avec un shell. D'autre part, il est souvent plus efficace de taper une simple commande dans un shell que de rechercher un outil graphique et de parcourir les divers menus, puis de choisir les options de la commande désirée avant de valider. Des ergonomes ont démontré, et des graphistes du monde entier le confirmeront, que la souris n'est pas le périphérique d'entrée le plus précis et le plus facile à utiliser pour manipuler les objets de l'environnement utilisateur. La plupart des programmeurs utilisent encore bon nombre de ce qu'on appelle des « raccourcis clavier » pour exécuter des commandes, même dans les environnements utilisateurs graphiques.**

**6°) Redirections des entrées/sorties**

**Pour pouvoir lancer un programme en arrière-plan, il est nécessaire qu'il n'ait pas besoin de demander des données à l'utilisateur. En effet, lorsqu'il est en arrière-plan, la saisie de ces données ne peut pas se faire, puisque le shell les interpréterait comme une nouvelle commande. De plus, tout affichage en provenance d'une commande en arrière-plan apparaît sur la console tel quel, et risque de se mélanger avec l'affichage des autres programmes ou même avec la commande en cours d'édition. C'est pour résoudre ces problèmes que le mécanisme des redirections a été introduit.**

**Le mécanisme des *redirections* a pour but de transférer les données provenant d'un flux vers les données d'un autre flux. Il se base sur la notion de descripteur de fichier, utilisée par la plupart des systèmes Unix. Un *descripteur de fichier* est un numéro utilisé par les programmes pour identifier les fichiers ouverts. Les descripteurs 0, 1 et 2 sont respectivement affectés d'office au flux d'entrée standard (nommé « stdin »), au flux de sortie standard (« stdout ») et au flux d'erreur standard (« stderr »), qui en général apparaît également sur l'écran.**

**Les descripteurs de fichiers d'un processus sont généralement hérités par tous ses processus fils. Cela signifie que, lors de leur lancement, ces processus peuvent utiliser tous les descripteurs de fichiers mis à leur disposition par leur père. Dans le cas des lignes de commande, les processus fils sont les processus lancés par le shell, et les descripteurs de fichiers hérités sont donc les descripteurs de fichiers du shell. C'est de cette manière que le shell peut manipuler les descripteurs de fichiers des processus qu'il lance : il effectue d'abord les redirections sur ses propres descripteurs de fichiers, puis il lance le processus fils avec ces redirections actives. Le mécanisme est donc complètement transparent pour les processus fils.**

**Le mécanisme des redirections permet en fait d'injecter dans un descripteur de fichier des données provenant d'un autre descripteur ou d'un fichier identifié par son nom, et d'envoyer les données provenant d'un descripteur de fichier dans un autre descripteur ou dans un fichier identifié par son nom. Si l'on utilise les descripteurs de fichiers des flux d'entrée / sortie standards, on peut exécuter n'importe quelle commande interactive en arrière-plan**.

1. **Redirections de données en entrée**

**Pour injecter des données provenant d'un fichier dans le descripteur de fichier n d'un processus, il suffit d'ajouter la ligne suivante à la fin de la commande permettant de lancer ce processus :**

**n<fichier**

**où fichier est le nom du fichier dont les données doivent être injectées dans le descripteur n. Dans cette syntaxe, le descripteur peut ne pas être précisé. Dans ce cas, le shell utilisera le descripteur 0, et les données du fichier seront donc envoyées dans le flux d'entrée standard du processus. Par exemple, supposons que l'on désire utiliser une commande nommée « search », et que cette commande demande un certain nombre d'informations lors de son exécution. Si l'on sait à l'avance les réponses aux questions qui vont être posées, on peut créer un fichier de réponse (nommé par exemple « answer.txt ») et alimenter la commande « search » avec ce fichier. Pour cela, on utilisera la ligne de commande suivante :**

**search < answer.txt**

**Il est également possible d'injecter des données provenant d'un autre descripteur de fichier dans un descripteur de fichier. On utilisera pour cela la syntaxe suivante :**

**n<&s**

**où n est toujours le descripteur de fichier du processus à exécuter dans lequel les données doivent être injectées, et s est un descripteur de fichier contenant les données sources à injecter. Par défaut, si n n'est pas précisé, le flux d'entrée standard du processus sera utilisé.**

1. **Redirection de données en sortie**

**Inversement, il est possible d'enregistrer les données écrites par un processus dans un de ses descripteurs de fichier dans un fichier. Pour cela, on utilisera l'opérateur '>' avec la syntaxe suivante :**

**n>fichier**

**où n est le numéro du descripteur de fichier du processus à enregistrer, et fichier est le nom du fichier dans lequel les données doivent être stockées. Par défaut, si n n'est pas spécifié, le descripteur du flux de sortie standard sera utilisé (descripteur 1). Par exemple, si la commande précédente affiche des résultats et que l'on désire les stocker dans le fichier « result.txt », on utilisera la ligne de commande suivante :**

**search < answer.txt >result.txt**

**Notez que cette commande détruira systématiquement le contenu du fichier « result.txt » et le remplacera par les informations provenant du flux de sortie standard du processus « search ». Il est possible de ne pas vider le fichier « result.txt » et d'ajouter les informations en fin de fichier, en utilisant l'opérateur '>>' à la place de l'opérateur '>'. Ainsi, la commande suivante :**

**search < answer.txt >>result.txt**

**aura pour effet d'ajouter à la fin du fichier « result.txt » les informations affichées par le processus « search ».**

**Le flux d'erreur standard, qui correspond normalement à l'écran et qui permet d'afficher les messages d'erreur, peut être redirigé avec l'opérateur '2>', de la même manière que l'opérateur '>' est utilisé pour le flux de sortie standard (puisque c'est le descripteur de fichier utilisé par défaut par l'opérateur '>'). Par exemple, si l'on veut envoyer les messages d'erreurs éventuels de la commande précédente vers le périphérique nul (c'est-à-dire le périphérique qui n'en fait rien) pour ignorer ces messages, on utilisera la ligne de commande suivante :**

**search <answer.txt >result.txt 2> /dev/null**

**Une telle ligne de commande est complètement autonome, et peut être lancée en arrière-plan, sans aucune intervention de l'utilisateur :**

**search <answer.txt >result.txt 2> /dev/null &**

**Il est également possible d'effectuer une redirection des données provenant d'un descripteur de fichier du processus vers un autre descripteur de fichier de ce processus. On utilisera pour cela la syntaxe suivante :**

**n>&d**

**où n est le descripteur de fichier dont les données doivent être redirigées, et d le descripteur de fichier destination. Cette syntaxe est souvent utilisée pour rediriger le flux d'erreur standard vers le flux d'entrée standard, lorsqu'on veut récupérer les erreurs et les messages d'exécution normale dans un même fichier. Par exemple, si l'on veut rediriger le flux de sortie et le flux d'erreurs de la commande « search » dans un même fichier, on utilisera la ligne de commande suivante :**

**search <answer.txt >result.txt 2>&1**

**Cette ligne de commande utilise deux redirections successives pour les données affichées par la commande « search » : la première redirige le flux de sortie standard vers un fichier, et la deuxième le flux d'erreur standard vers le flux de sortie standard. Notez que l'ordre des redirections est important. Elles sont appliquées de gauche à droite. Ainsi, dans la commande précédente, le flux de sortie standard est redirigé vers le fichier « result.txt », puis le flux d'erreur standard est injecté dans le flux de sortie standard ainsi redirigé.**

**Note : Il est également possible d'utiliser un autre descripteur de fichier que les descripteurs des flux standards. Cependant, il est nécessaire, dans ce cas, d'ouvrir ce descripteur dans le shell avant de lancer la commande. Cela peut se faire à l'aide de la syntaxe suivante :**

**n<>fichier**

**où n est un numéro de descripteur de fichier non encore utilisé, et fichier est un nom de fichier. Ce nouveau descripteur de fichier pourra être utilisé dans les commandes précédentes, afin de faire manipuler le fichier fichier par les processus fils de manière transparente.**

**Les descripteurs de fichiers ouverts de cette manière le restent d'une commande sur l'autre dans le shell. Cela implique que toutes les données écrites dans ces descripteurs de fichiers sont ajoutées automatiquement à la fin des fichiers manipulés par ces descripteurs. Ce comportement est différent de la redirection vers un fichier effectuée par l'opérateur '>', qui ouvre à chaque fois le fichier en écriture à son début et qui supprime donc toutes les données déjà existantes. Il n'y a donc pas d'opérateur '>>&' pour ajouter des données à un descripteur de fichier, car cela n'a pas de sens.**

**Les descripteurs de fichiers peuvent également être manipulés directement, par l'intermédiaire de fichiers virtuels du répertoire /dev/fd/. À chaque descripteur de fichier (y compris les descripteurs pour les flux d'entrée/sortie standards !) y correspond un fichier dont le nom est le numéro du descripteur. Par exemple, le fichier /dev/fd/2 correspond au flux d'erreur standard.**

**En fait, le répertoire /dev/fd/ est un lien symbolique vers le répertoire /proc/self/fd/ du système de fichiers virtuel /proc/. Ce système de fichiers est géré par le noyau directement, et permet d'accéder aux informations sur le système et les processus. Il contient en particulier un sous-répertoire portant le nom du PID de chaque processus existant dans le système, et chacun de ces répertoires contient lui-même un sous-répertoire fd/ où sont représentés les descripteurs de fichiers ouvert par le processus correspondant. Le système de fichiers /proc/ contient également un lien symbolique self/ pointant sur le sous-répertoire du processus qui cherche à l'ouvrir. Ainsi, /proc/self/fd/ est un chemin permettant à chaque processus d'accéder à ses propres descripteurs de fichiers.**

**En pratique, la manipulation directe des descripteurs de fichiers n'est réellement intéressante que pour les flux standards, dont les numéros de descripteurs sont fixes et connus de tous les programmes. Pour les autres descripteurs, cette technique est souvent inutilisable ou inutile, sauf lorsqu'on utilise des programmes sachant manipuler des descripteurs de numéros bien déterminés.**

**Un processus Unix possède par défaut trois voies d’interaction avec l’extérieur appelées entrées / sorties standard** **identifiées par un entier positif ou nul appelé** **descripteur de fichier.**

**En résumé :**

**Ces entrées / sorties standard sont :**

* **une entrée standard, stdin, avec un descripteur noté 0 (par défaut, c’est le clavier)**
* **une sortie standard, stdout, avec un descripteur noté 1 (par défaut, c’est l’écran)**
* **une sortie standard pour les messages d’erreurs, stderr, avec un descripteur noté 2 (par défaut, c’est l’écran).**

**Toute commande étant exécutée par un processus, nous dirons également qu’une commande possède trois entrées / sorties standard.**

**Pour ouvrir d'autres fichiers, il reste les descripteurs 3 à 9. Il est quelque fois utile d'affecter un de ces descripteurs supplémentaires de fichiers pour stdin, stdout ou stderr comme lien dupliqué temporaire. Ceci simplifie le retour à la normale après une redirection complexe et un remaniement.**

**Fermer les descripteurs de fichiers**

**n<&-**

**Ferme le descripteur de fichier *n*.**

**0<&-, <&-**

**Ferme stdin.**

**n>&-**

**Ferme le descripteur de fichiers de sortie *n*.**

**1>&-, >&-**

**Ferme stdout.**

**Exemples**

**Commande utilisant les entrées et sortie standard**

[root@machine1 /]#

[root@machine1 /]# **ls**

**bin boot cgroup dev etc home lib lib64 lost+found media misc mnt net opt proc root sbin selinux srv sys tmp usr var**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **ls w\***

**ls: impossible d'accéder à w\*: Aucun fichier ou dossier de ce type**

[root@machine1 /]#

**Sortie standards redirigées**

1. **Sortie du bon résultat dans /tmp/sortie et des erreurs sur la sortie standard**

root@machine1 /]#

[root@machine1 /]# **ls w\* 1>/tmp/sortie**

**ls: impossible d'accéder à w\*: Aucun fichier ou dossier de ce type**

[root@machine1 /]# **more /tmp/sortie**

[root@machine1 /]#

1. **Sortie des erreurs dans un fichier et le bon résultat sur la sortie standard**

[root@machine1 /]#

[root@machine1 /]# **ls w\* 2>/tmp/erreur**

[root@machine1 /]# **more /tmp/erreur**

**ls: impossible d'accéder à w\*: Aucun fichier ou dossier de ce type**

[root@machine1 /]#

1. **Sortie du résultat et des erreurs dans deux fichiers différents**

[root@machine1 /]#

[root@machine1 /]# **ps -ax | head -5**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

**PID TTY STAT TIME COMMAND**

**1 ? Ss 0:01 /sbin/init**

**2 ? S 0:00 [kthreadd]**

**3 ? S 0:00 [migration/0]**

**4 ? S 0:00 [ksoftirqd/0]**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **ps -ax 1>/tmp/resultat 2>/tmp/erreur**

[root@machine1 /]#

[root@machine1 /]# **tail -5 /tmp/resultat**

**2676 ? Sl 0:05 gnome-terminal**

**2677 ? S 0:00 gnome-pty-helper**

**2678 pts/0 Ss 0:00 bash**

**18321 ? S 0:00 pickup -l -t fifo -u**

**28352 pts/0 R+ 0:00 ps -ax**

[root@machine1 /]#

[root@machine1 /]# **more /tmp/erreur**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

[root@machine1 /]#

1. **Redirection des sorties (résultat et erreur) dans le même fichier**

[root@machine1 /]#

[root@machine1 /]# **ps -ax 1>>/tmp/resul 2>&1**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **head -5 /tmp/resul**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

**PID TTY STAT TIME COMMAND**

**1 ? Ss 0:01 /sbin/init**

**2 ? S 0:00 [kthreadd]**

**3 ? S 0:00 [migration/0]**

[root@machine1 /]#

1. **Redirection dans un fichier /dev/null.**

**Le fichier spécial /dev/null est appelé « poubelle » ou « puits » car toute sortie qui y est redirigée, est perdue. En général, il est utilisé lorsqu’on est davantage intéressé par le code de retour de la commande plutôt que par les résultats ou messages d’erreur qu’elle engendre.**

[root@machine1 /]# **ps -ax 2>/dev/null | head -5**

**PID TTY STAT TIME COMMAND**

**1 ? Ss 0:01 /sbin/init**

**2 ? S 0:00 [kthreadd]**

**3 ? S 0:00 [migration/0]**

**4 ? S 0:00 [ksoftirqd/0]**

[root@machine1 /]#

**Attention, lorsque la redirection se fait dans un fichier par le caractère ‘>’, le fichier sera créé s’il n’existe pas, sinon il sera écrasé.**

**Si nous utilisons le caractère double ‘>>’, le fichier sera créé s’il n’existe pas sinon, les informations seront envoyées à la suite du fichier.**

1. **Redirection par les tubes.**

**Les redirections sont très pratiques lorsqu'il s'agit d'injecter un fichier dans le flux d'entrée standard d'un processus, ou inversement de rediriger le flux standard d'une commande vers un fichier, mais elles ont justement le défaut de devoir utiliser des fichiers. Il est des situations où l'on désirerait injecter le résultat d'une commande dans le flux d'entrée standard d'une autre commande, sans passer par un fichier intermédiaire. Cela est heureusement réalisable, grâce à ce que l'on appelle les « tubes ».**

**Pour rediriger le résultat d'une commande dans le flux d'entrée d'une autre commande, il faut utiliser l'opérateur '|'. Cet opérateur représente un tuyau canalisant les données issues d'une commande vers le flux d'entrée standard de la commande suivante, d'où le nom de « pipe » en anglais (ce qui signifie « tuyau » ou « tube »). L'opérateur tube s'utilise de la manière suivante :**

* **on écrit la première commande, qui doit fournir les données à la deuxième commande ;**
* **on écrit l'opérateur tube ;**
* **on écrit la deuxième commande, qui doit lire les données provenant de la première.**

**La commande se trouvant à la gauche de l'opérateur tube doit être complète, avec ses autres redirections éventuelles. La redirection dans un tube s'effectue après les autres types de redirections vues précédemment.**

**Le système contrôle l'exécution des processus qui se trouvent aux deux bouts d'un tube, de telle sorte que le transfert de données puisse toujours se faire. Si le processus source a trop de données, il est figé par le système d'exploitation en attendant que le processus consommateur ait fini de traiter les données déjà présentes. Inversement, si le processus source est trop lent, c'est le processus consommateur qui attendra patiemment que les données soient disponibles.**

**Les tubes sont utilisés très couramment, ne serait-ce que pour afficher page par page le contenu d'un répertoire. La commande suivante effectue un tel travail :**

**ls | less**

**Ici, le résultat de la commande ls est redirigé vers la commande less, qui permet d'afficher page par page (et de revenir en arrière dans ces pages) la liste des fichiers du répertoire courant.**

**Un autre exemple pratique est le déplacement de toute une arborescence de fichiers d'un système de fichiers à un autre. Vous ne pourrez pas y parvenir à l'aide de la commande mv, car celle-ci ne fait que modifier la structure du système de fichiers pour déplacer les fichiers et les répertoires, elle ne peut donc pas fonctionner avec deux systèmes de fichiers. Vous ne pouvez pas non plus utiliser la commande cp, car celle-ci ne prendra pas en compte les dates des fichiers, leur propriétaire et leur groupe, ainsi que les liens symboliques et physiques. Il faut donc impérativement utiliser un programme d'archivage. La méthode à suivre est donc de créer une archive temporaire, puis de se déplacer dans le répertoire destination, et enfin d'extraire l'arborescence de l'archive :**

**cd source**

**tar cvf archive.tar \***

**cd destination**

**tar xvf source/archive.tar**

**rm source/archive.tar**

**Malheureusement, cette technique nécessite beaucoup de place disque, puisque l'archive temporaire est stockée directement sur disque. De plus, elle est assez lente, car toutes les données à copier sont recopiées sur le disque dur, et relues ensuite, pour finalement être détruites... La vraie solution est de réaliser un tube entre les deux processus tar invoqués. Dans ce cas, le transfert se fait simplement via la mémoire vive :**

**cd source**

**tar cv \* | (cd destination ; tar xvf -)**

**La commande à utiliser est cette fois un peu plus compliquée, car la commande d'extraction des fichiers nécessite un changement de répertoire. Il faut donc utiliser une commande multiple du shell. Ces commandes sont constituées de plusieurs autres commandes séparées par des points virgules. La première commande effectuée ici est le changement de répertoire, et la deuxième est l'extraction par tar de l'archive qui lui est transférée par le flux d'entrée standard (représenté ici par '-'). Ces deux commandes sont mises entre parenthèses, car l'opérateur '|' du tube est prioritaire sur l'opérateur ';' de concaténation des commandes du shell. Si vous trouvez que cela est un peu compliqué, je vous l'accorde. Cependant, la commande qui utilise le tube consomme deux fois moins d'espace disque et est deux fois plus rapide que la commande qui n'en utilise pas. Je vous invite à mesurer le gain de temps sur un répertoire contenant un grand nombre de données (utilisez la commande time !).**

**Les tubes nommés**

**Les tubes créés par l'opérateur '|' constituent ce que l'on appelle des tubes anonymes, car ils sont créés directement par le shell pour une commande donnée. Il est possible de créer manuellement des tubes en leur donnant un nom, et de les utiliser a posteriori dans plusieurs commandes. Ces tubes constituent ce que l'on appelle des tubes nommés.**

**En fait, les tubes nommés sont des fichiers spéciaux, que l'on crée dans un système de fichiers capable de les gérer. Les seules opérations réalisables sont l'écriture et la lecture, sachant que les données écrites en premier seront forcément les premières données lues. C'est ce comportement qui a donné leur nom à ces fichiers, que l'on appelle des « FIFO » (abréviation de l'anglais « First In First Out »). De plus, la quantité de données en transit dans ces fichiers est souvent très réduite, ce qui fait que ces données sont toujours placées dans la mémoire cache du système. Ainsi, bien qu'il s'agisse de fichiers, aucune écriture ou lecture sur disque n'a lieu lors de l'utilisation d'un pipe.**

**Les tubes nommés sont créés par la commande mkfifo, dont la syntaxe est la suivante :**

**mkfifo nom**

**où nom est le nom du tube nommé. Notez que cette commande échouera sur les systèmes de fichiers incapables de gérer les tubes nommés.**

**Une fois créé, le fichier de tube peut être utilisé comme n'importe quel fichier dans les redirections que l'on a vues dans la section précédente. Par exemple, la redirection suivante :**

**ls | less**

**peut être réécrite pour utiliser un tube nommé temporaire de la manière suivante :**

**mkfifo /tmp/tempfifo**

**ls > /tmp/tempfifo**

**less < /tmp/tempfifo**

**La destruction d'un tube nommé se fait comme n'importe quel fichier, à l'aide de la commande rm.**

**La commande tee**

**La commande tee est un petit programme permettant d'enregistrer les données qu'il reçoit dans son flux d'entrée standard dans un fichier et de les renvoyer simultanément vers son flux de sortie standard. Elle est couramment utilisée, en conjonction avec les tubes, pour dupliquer un flux de données. Sa syntaxe est la suivante :**

**tee fichier**

**où fichier est le nom du fichier dans lequel le flux d'entrée standard doit être enregistré.**

**Supposons par exemple que l'on désire rediriger tous les messages (d'erreur ou non) de la commande ls /proc/1/\* dans un fichier result.txt, tout en continuant à les visualiser sur l'écran. Pour cela, on utilisera la commande suivante :**

**ls -l /proc/1 2>&1 | tee result.txt**

**À l'issue de cette commande, le fichier result.txt contiendra une copie des données qui ont été émises par la commande** **ls -l /proc/1 2>&1**.

**La commande xargs**

**La commande xargs permet d'appeler une autre commande, en passant en paramètre les données qu'elle reçoit dans le flux d'entrée standard. Sa syntaxe est la suivante :**

**xargs commande**

**où commande est la commande que xargs doit exécuter. xargs construira une ligne de commande complète pour cette commande, en utilisant comme paramètres les données issues du flux d'entrée standard. Une fois cette ligne de commande construite, xargs l'exécutera. Par exemple, la commande suivante :**

**ls -l**

**peut être exécutée également de la manière suivante :**

**xargs ls**

**et en tapant la chaîne de caractères « -l » suivie du caractère de fin de fichier CTRL+D.**

**La commande xargs est une commande extrêmement utile lorsqu'elle est utilisée conjointement avec les tubes, parce qu'elle permet d'utiliser le résultat d'une commande en tant que paramètre pour une autre commande. Ce mécanisme est donc complémentaire de celui des pipes, puisque ceux-ci permettaient d'utiliser le résultat d'une commande pour alimenter le flux d'entrée standard d'une autre commande.**

**Un exemple plus utile que le précédent permettra de mieux comprendre comment on utilise la commande xargs. Supposons que l'on désire trouver tous les fichiers d'une arborescence complète dont l'extension est .txt et contenant la chaîne de caractères « test ». La liste des fichiers de l'arborescence peut être déterminée simplement à l'aide de la commande find, et la recherche du texte dans les fichiers se fait naturellement à l'aide de la commande grep. On utilisera xargs pour construire la ligne de commande pour grep, à partir du résultat fourni par la commande find :**

**find -name "\*.txt" | xargs grep -l "test"**

**Cette commande est plus simple et plus efficace que la commande équivalente :**

**find -name "\*.txt" -exec grep -l "test" {} \;**

parce que **grep** n'est exécuté qu'une seule fois (alors que l'option -exec de la commande **find** l'exécute pour chaque fichier trouvé).

**7) Commandes utiles sur les processus**.

**a) Informations sur les processus actifs :** **la commande ps**

syntaxe : **ps [options**]

La commande permet d'afficher des informations sur les processus qui appartiennent à la même session que le processus **shell** depuis lequel la commande est lancée.

**Commande ps m’affichant uniquement les processus lancés dans ma session.**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **whoami**

**jmm**

[jmm@machine1 ~]$ **ps**

**PID TTY TIME CMD**

**1375 pts/0 00:00:00 bash**

**1426 pts/0 00:00:00 ps**

[jmm@machine1 ~]$

**Avec l'option "–l" et "–f" nous avons une liste plus complète de renseignements.**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **ps -axlf | more**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

**F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND**

**1 0 2 0 20 0 0 0 kthrea S ? 0:00 [kthreadd]**

**1 0 3 2 -100 - 0 0 migrat S ? 0:00 \\_ [migration/0]**

**1 0 4 2 20 0 0 0 ksofti S ? 0:00 \\_ [ksoftirqd/0]**

**1 0 5 2 -100 - 0 0 cpu\_st S ? 0:00 \\_ [migration/0]**

**5 0 6 2 -100 - 0 0 watchd S ? 0:00 \\_ [watchdog/0]**

**1 0 7 2 20 0 0 0 worker S ? 0:01 \\_ [events/0]**

**1 0 8 2 20 0 0 0 worker S ? 0:00 \\_ [cgroup]**

**1 0 9 2 20 0 0 0 worker S ? 0:00 \\_ [khelper]**

**…**

**b) la commande kill**

**kill [-signaux] numero\_processus**

**Dans un système Unix, tous les processus peuvent recevoir des messages, envoyés soit par l'utilisateur, soit par un autre processus, soit par le système. Ces messages sont appelés *signaux*. La plupart des signaux sont envoyés par le système pour indiquer au processus qu'il a fait une faute et qu'il va être terminé.**

**La commande kill permet d'envoyer un signal à un processus. Le ou les processus sont identifiés par leur numéro d'identification qu'il est possible de connaître à la commande "ps".**

**Seul l'administrateur a le droit de tuer des processus dont il n'est pas le propriétaire.**

**Le signal SIGKILL, de numéro " 9", garanti l'arrêt du processus.**

**kill –9 0 met fin à tous les processus d'une session, même ceux lancé en arrière plan.**

| **Numéro de signal** | **Signification** |
| --- | --- |
| 15 | Signal de terminaison de processus. |
| 9 | Signal de destruction inconditionnelle de processus. |
| 19 | Signal de suspension de processus. |
| 18 | Signal de reprise d'exécution d'un processus suspendu. |

**Lorsqu'aucun signal n'est spécifié, le signal 15 de terminaison est utilisé par défaut. Ce signal demande au processus en cours d'exécution de se terminer immédiatement. Il peut être capté par le processus, pour lui donner une chance d'enregistrer les données sur lesquelles il travaillait et de libérer les ressources qu'il utilisait. Pour certains processus, cela ne fonctionne pas, et il faut utiliser le signal de destruction du processus à l'aide de la commande suivante :**

**kill -9 PID**

**Attention cependant à cette commande : le processus est immédiatement détruit, sans autre forme de procès. Il peut donc s'ensuivre une perte de données, n'en abusez donc pas trop.**

**c) processus en arrière-plan**

**Lorsque l'utilisateur tape une commande, le shell crée un processus fils, le shell se met en attente de fin d'exécution de ce processus; c'est-à-dire que le processus shell est endormi et l'utilisateur n'a plus la main jusqu'à la fin de la commande.**

**Il est possible de ne pas endormir le processus shell et donc de pouvoir continuer à travailler simultanément. pour cela, il faut lancer la commande en arrière plan.**

**Le symbole d'exécution en arrière plan est le caractère "&" qui doit être ajoutée en fin de commande après les options des paramètres.**

**Exemple : sort grosfich > nouvfich &**

**./programme\_calcul\_paie &**

**Le système répond en donnant le numéro du processus PID ainsi créé, puis le prompt indiquant que le shell est à nouveau en attente de commande. L'utilisateur n'est pas averti de la fin du processus ainsi créé et si celui-ci doit afficher des messages à l'écran, il y aura affichage de ces sorties. L'utilisateur aura donc intérêt à rediriger les sorties d'une commande différée.**

**d) la commande nohup**

L'utilitaire **nohup** permet à la commande passée en paramètre de continuer son exécution même si l'utilisateur se déconnecte du système (**nohup : no hangup**, pas d'arrêt imprévu).

Si une demande de redirection de la sortie standard n'a pas été formulée, celle-ci et la sortie erreur standard sont automatiquement redirigées sur le fichier **nohup.out** du répertoire de travail.

# Exemple

**nohup sort grosfich > nouvfich &**

**e) suspension d’un processus**

Si un processus est exécuté en avant plan, il est possible de le suspendre en frappant simultanément sur les touches **Ctrl -Z**. Un processus suspendu peut être redémarré en avant-plan via la commande **fg** (pour foreground) ou en arrière plan avec la commande **bg** (pour background).

**f) La commande nice.**

**Vous disposez de la commande nice pour modifier la priorité d'un processus. La syntaxe est la suivante :**

**nice -valeur commande**

**Plus le nombre est grand, plus la priorité est faible. Par exemple une valeur de 0 donne, la priorité la plus haute 19 donne la priorité la plus faible.**

**La fourchette de valeur dépend de l'UNIX qu'on utilise.**

**Par exemple :**

**nice -5 ps -ef**

**Généralement on utilise nice sur des commandes qui prennent du temps, sur des commandes courantes l'effet de nice est imperceptible. On l'utilisera par exemple pour compiler un programme.**

**nice -5 cc monprogramme.c**

**g) La commande pstree**

**pstree affiche les processus en cours d’exécution sous la forme d’un arbre. La racine de l’arbre est soit un processus dont nous avons donné le pid, soit init si le pid est omis. Si un nom d’utilisateur est fourni, tous les arbres de processus prenant racine sur des processus appartenant à cet utilisateur seront affichés.**

**pstree fusionnera visuellement les branches identiques en les mettant entre crochets et en les préfixant par le nombre de répétitions.**

**Par exemple,**

**init-+-getty**

**|-getty**

**|-getty**

**‘-getty**

**devient**

**init---4\*[getty]**

**OPTIONS**

**-a Afficher les arguments de la ligne de commande. Si la ligne de commande correspond à un processus endormi en zone de swap, ce processus est affiché entre parenthèses. L’option -a désactive implicitement le compactage des branches.**

**-A Utiliser les caractères ASCII pour dessiner l’arbre.**

**-c Désactiver le compactage des sous-arbres identiques. Par défaut, les sous-arbres sont compactés chaque fois que c’est possi-**

**ble.**

**- G Utiliser les caractères VT100 pour dessiner l’arbre.**

**-h Mettre en évidence (Ndt : highlight) le processus courant et ces ancètres. Cela devient une non opération si le terminal ne gère pas la mise en évidence ou si ni le processus courant, ni l’un de ces ancètres ne se trouvent dans le sous-arbre à afficher.**

**-H Comme -h, mais mettre en évidence le processus spécifié à la place. Contrairement à -h, pstree échouera si la mise en évidence**

**n’est pas disponible.**

**-l Afficher des lignes longues. Par défaut, les lignes sont tronquées à la largeur d’affichage ou à 132 caractères si la sortie ne**

**se fait pas sur un terminal ou si la largeur d’affichage est inconnue.**

**-n Trier par PID les processus ayant le même ancètre plutôt que par nom. (Tri numérique.)**

**-p Afficher les PID. Ils le sont sous forme décimale entre parenthèses après chaque nom de processus. L’option -p désactive**

**implicitement le compactage.**

**-u Afficher les transitions d’UID. Lorsque l’UID d’un processus diffère de l’UID de son père, le nouvel UID est affiché entre par-**

**enthèses après le nom du processus.**

**-U Utiliser les caractères UTF-8 pour dessiner l’arbre. Sous Linux 1.1-54 et précédents, on entre en mode UTF-8 sur la console avec echo -e ’ 33%8’ et on le quitte avec echo -e ’ 33%@’**

**-V Afficher les informations de version.**

**-Z (SELinux) Afficher le contexte de sécurité pour chaque processus.**

**Exemple**

[root@machine1 ~]# **pstree**

**init─┬─NetworkManager**

**├─abrtd**

**├─acpid**

**├─atd**

**├─auditd─┬─audispd─┬─sedispatch**

**│ │ └─{audispd}**

**│ └─{auditd}**

**├─automount───4\*[{automount}]**

**├─bonobo-activati───{bonobo-activat}**

**├─certmonger**

**├─clock-applet**

**├─console-kit-dae───63\*[{console-kit-da}]**

**├─crond**

**├─cupsd**

**├─2\*[dbus-daemon───{dbus-daemon}]**

**├─2\*[dbus-launch]**

**├─devkit-power-da**

**├─gconf-im-settin**

**├─gconfd-2**

**├─gdm-binary─┬─gdm-simple-slav─┬─Xorg**

**│ │ ├─gdm-session-wor─┬─gnome-session─┬─abrt-applet**

**│ │ │ │ ├─gdu-notificatio**

**│ │ │ │ ├─gnome-panel**

**│ │ │ │ ├─gnome-power-man**

**│ │ │ │ ├─gpk-update-icon**

**│ │ │ │ ├─metacity───{metacity}**

**│ │ │ │ ├─nautilus**

**│ │ │ │ ├─nm-applet**

**│ │ │ │ ├─polkit-gnome-au**

**│ │ │ │ ├─seapplet**

**│ │ │ │ └─{gnome-session}**

**│ │ │ └─{gdm-session-wo}**

**│ │ └─{gdm-simple-sla}**

**│ └─{gdm-binary}**

**├─gdm-user-switch**

**├─gnome-keyring-d───2\*[{gnome-keyring-}]**

**├─gnome-screensav**

**├─gnome-settings-───{gnome-settings}**

**├─gnome-terminal─┬─bash───pstree**

**│ ├─gnome-pty-helpe**

**│ └─{gnome-terminal}**

**├─gvfs-fuse-daemo───3\*[{gvfs-fuse-daem}]**

**├─gvfs-gdu-volume**

**├─gvfsd**

**├─gvfsd-trash**

**├─hald─┬─hald-runner─┬─hald-addon-acpi**

**│ │ ├─hald-addon-inpu**

**│ │ └─hald-addon-stor**

**│ └─{hald}**

**├─im-settings-dae**

**├─master─┬─pickup**

**│ └─qmgr**

**├─mcelog**

**├─5\*[mingetty]**

**├─modem-manager**

**├─notification-ar**

**├─notification-da**

**├─pcscd───{pcscd}**

**├─polkitd**

**├─portreserve**

**├─pulseaudio───2\*[{pulseaudio}]**

**├─restorecond**

**├─rpc.statd**

**├─rpcbind**

**├─rsyslogd───3\*[{rsyslogd}]**

**├─rtkit-daemon───2\*[{rtkit-daemon}]**

**├─seahorse-agent**

**├─seahorse-daemon**

**├─sshd**

**├─tpvmlp**

**├─trashapplet**

**├─udevd───2\*[udevd]**

**├─udisks-daemon─┬─udisks-daemon**

**│ └─{udisks-daemon}**

**├─uuidd**

**├─2\*[vmtoolsd]**

**├─vmware-vmblock-───2\*[{vmware-vmblock}]**

**├─wnck-applet**

**├─wpa\_supplicant**

**└─xinetd**

**h) la commande w**

**w affiche des informations sur les utilisateurs actuellement sur la machine, et leur processus. L’en-tête montre, dans cet ordre, l’heure actuelle, depuis quand le système est en route, combien d’utilisateurs sont actuellement connectés, et la moyenne de la charge processeur pour les dernières 1, 5 et 15 minutes.**

**Exemple**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **who**

**jmm tty7 2014-06-02 10:32 (:0)**

**jmm pts/0 2014-06-02 10:32 (:0.0)**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **whoami**

**jmm**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **w**

**10:33:08 up 26 min, 2 users, load average: 0,67, 0,17, 0,06**

**USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT**

**jmm tty7 :0 10:32 26:20 1.36s 0.14s pam: gdm-passwo**

**jmm pts/0 :0.0 10:32 0.00s 0.00s 0.00s w**

[jmm@machine1 ~]$

**i) la commande pgrep, pwdx**

**Pour connaitre le PID d’un processus, on peut utiliser la commande pgrep.**

**Pour connaitre le répertoire de travail d’un processus, on peut utiliser la commande pwdx**

[root@machine1 ~]#

[root@machine1 ~]# **pgrep cron**

**1966**

[root@machine1 ~]# **pwdx 1966**

**1966: /**

[root@machine1 ~]#

**8) Commandes suivantes sont etudiées dans le chapitre shell**:

**regroupements des commandes** : **(c1; c2; c3); {c1;c2;c3}**

**exécution conditionnelle** : **c1 && c2 (c1 et c2) c1|| c2 (c1 ou c2)**

**communication : c1 | c2**

**arrière plan** : **c&**

**redirection**: **>, <, >>**

**sortir d'un processus** : **exit**

**endormir un processus** : **sleep**

**9) Variables générées automatiquement par le shell** :

**$?** **valeur de retour de la dernière commande,**

**$!** **numéro du dernier processus ( PID ) lancé en arrière- plan,**

**$$** **numéro de processus ( PID ) en cours d’ éxécution,**

**$#** **nombre de paramètres transmis,**

**$\*** **ou** **$@**  **liste des paramètres,**

**$0** **nom de la commande,**

$1 **premier paramètre de la commande,**

**$n** **n-ième paramètre de la commande.**

**10) Illustration des variables générées automatiquement par le shell**

[root@machine1 bin]#

[root@machine1 bin]# **more param.sh**

**echo -e "\n\n"**

**echo -e " Le nombre de parametres du script est : $# \n"**

**echo -e " La liste des parametres du scrit est : $\* \n"**

**echo -e " Le nom du script actuellement lance est : $0 \n"**

**echo -e " Le premier parametre est : $1 \n"**

**echo -e "\n Nous allons interroger le systeme pour connaitre le PID de ce script\n"**

**ps -ax | grep param.sh**

**echo -e "\n Le PID du processus en cours est: $$\n"**

[root@machine1 bin]#

**Lancement du script param.sh**

[root@machine1 bin]#

[root@machine1 bin]# **sh param.sh iut 1 4 6**

**Le nombre de parametres du script est : 4**

**La liste des parametres du script est : iut 1 4 6**

**Le nom du script actuellement lance est : param.sh**

**Le premier parametre est : iut**

**Nous allons interroger le systeme pour connaitre le PID de ce script**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

**14930 pts/0 S+ 0:00 sh param.sh iut 1 4 6**

**14932 pts/0 S+ 0:00 grep param.sh**

**Le PID du processus en cours est: 14930**

[root@machine1 bin]#

**CRON**

**1°) Généralités.**

**cron est un processus activé comme un deamon permettant de lancer des commandes différées. Ces commandes sont décrites dans des fichiers ‘crontab’.**

**Le fichier /etc/crontab et les fichiers présents dans /etc/cron.d sont gérés par l’administrateur système. Par mesure de sécurité, ils doivent être accédés que par root, et devront avoir un droit 600.**

**Les fichiers présents dans /var/spool/cron sont des fichiers crontab ayant des noms existants dans /etc/passwd ou dans un annuaire.**

**Les fichiers trouvés sont chargés en mémoire au moment du lancement du démon crond. Ce lancement se fait par la commande ‘service crond start’**

**Ensuite, cron s’éveille toutes les minutes, examine les crontabs mémorisés, et vérifie chaque commande pour savoir s’il doit la lancer dans la minute à venir. Lors de l’exécution d’une commande, toute sortie est envoyée par mail au propriétaire de la crontab (ou à l’utilisateur dont le nom est mentionné dans la variable d’environnement MAILTO si elle existe). De plus, cron vérifie chaque minute si la date de modification de son répertoire de stockage (ainsi que la date de /etc/crontab) a été modifiée. Si c’est le cas, cron examinera les dates de modifications de chaque fichier crontab, et rechargera ceux qui ont été modifiés. Ainsi, cron n’a pas besoin d’être redémarré après la modification d’un fichier crontab. Notez que la commande crontab met**

**à jour la date de modification du répertoire de stockage si un changement a lieu.**

**2°) Format d’un fichier crontab**.

**Un fichier crontab contient des instructions pour le démon cron(8). Ces instructions ont la forme générale suivante : « lancer cette commande à telle heure, tel jour ». Chaque utilisateur dispose de sa propre table crontab, et les commandes contenues dans une table seront exécutées sous l’identifiant du propriétaire de la table.**

**Les lignes blanches, et les espaces et tabulations en tête de lignes sont ignorées. Les lignes dont le premier caractère non-blanc est un dièse « # » sont considérées comme des commentaires, et sont également ignorées. Notez que les commentaires ne peuvent pas se trouver sur la même ligne qu’une commande cron, car ils seraient considérés comme des arguments en ligne pour la commande à invoquer. De même, un commentaire ne peut pas se trouver sur la même ligne qu’une affectation de variable d’environnement.**

**Une ligne active dans une crontab devra donc être soit une affectation de variable d’environnement, soit une commande cron**

1. **Variables d’environnement.**

**Une ligne d’affectation d’environnement est de la forme**

**nom = valeur**

**où les espaces autour du signe égal « = » sont facultatifs, et où tous les espaces ultérieurs feront partie intégrante de la valeur affectée à la variable ayant le nom indiqué. La chaîne de valeur peut être inscrite entre guillemets ou entre apostrophes afin de protéger les blancs initiaux et finaux.**

**Plusieurs variables d’environnement sont automatiquement configurées par le démon cron(8). SHELL est remplie avec /bin/sh, LOGNAME et HOME sont configurées à partir de la ligne de /etc/passwd correspondant au propriétaire de la crontab. HOME et SHELL peuvent être surchargés par des affectations explicites dans la crontab, LOGNAME non. La variable MAILTO s’il doit envoyer le résultat d’une commande exécutée dans « cette » crontab. Si MAILTO est définie (et non vide), le résultat est envoyé à l’utilisateur indiqué. Si MAILTO est défini et vide (MAILTO=""), aucun courrier ne sera envoyé. Sinon, le courrier sera émis vers le propriétaire de la crontab**

1. **Format d’une commande cron**

**Chaque ligne dispose de 5 champs de date et heure, suivi d’un nom d’utilisateur s’il s’agit du fichier crontab central du système, suivi d’une commande. Notez bien que la syntaxe diffère entre un fichier /etc/crontab, les fichiers présents dans /etc/cron.d et ceux présents dans /var/spool/cron. En effet, le champ ‘utilisateur’ n’est pas présent dans le fichier crontab défini par un utilisateur lui-même**. **Cron examine ses données une fois par minute.**

**Les cinq premiers champs sont les champs de date et heure :**

champs valeurs autorisées

----- --------------

minute 0-59

heure 0-23

jour du mois 0-31

mois 0-12 (ou noms, voir plus bas)

jour de semaine0-7 (0 et 7 sont Dimanche, ou les noms)

**Un champ peut contenir un astérisque « \* », qui correspond à l’intervalle « premier-dernier »**

**Les intervalles de nombres sont permis. Ils se présentent sous forme de deux nombres séparés par un tiret. Les bornes sont inclues.**

**Par exemple, l’intervalle horaire 8-11 correspond à une exécution aux heures 8, 9, 10, et 11.**

**Les listes sont permises. Une liste est une série de nombres ou d’intervalles séparés par des virgules. Exemple « 1,2,5,9 »,« 0-4,8-12 ». Des valeurs de « pas » peuvent être associées aux intervalles. À la suite d’un intervalle, un « /<nombre> » précise le pas à adopter pour parcourir l’intervalle. Par exemple « 0-23/2 » dans le champ horaire demande une exécution toutes les heures paires. Les pas sont également autorisés à la suite d’un astérisque, ainsi, pour dire « toutes les deux heures », on peut utiliser simplement « \*/2 ».**

**On peut employer les noms des mois ou des jours de la semaine, en utilisant les trois premières lettres (pas de différence majuscule/minuscule). (Ndt : en anglais !) Les intervalles, ou les listes de noms ne sont pas acceptés.**

**Le sixième champ pour une crontab système (/etc/crontab ou fichiers de /etc/cron.d) indique l’utilisateur qui doit lancer la commande (en général, root)**

**Le sixième champ pour une crontab ‘utilisateur’ (qui se trouve dans /var/spool/cron) ou septième champ pour une crontab système**  **(le reste de la ligne)**

**indique la commande à exécuter. Tout le reste de la ligne, jusqu’au retour chariot ou au caractère « % », sera exécuté par /bin/sh, ou par le shell mentionné dans la variable SHELL du fichier cron. Les signes pourcentages « % » dans les commandes seront transformés en retour-chariot, sauf s’ils sont précédés par un backslash « \ ». Toutes les données se trouvant à la suite du premier % seront transmises à la commande sur son entrée standard.**

**Note : Le jour d’exécution d’une commande peut être spécifié par deux champs (jour du mois, et jour de la semaine). Si les deux champs sont remplis (c.-à-d. pas \*), la commande sera lancée quand l’un des champs correspond à la date en cours.**

**Par exemple :**

**« 30 4 1,15 \* 5 » exécutera une commande à 4:30 du matin les 1er et 15 de chaque mois, ET chaque Vendredi**.

**EXEMPLE DE FICHIER CRONTAB**

**# utiliser /bin/sh pour lancer les commandes, quoiqu’en dise /etc/passwd.**

**SHELL=/bin/sh**

**# Envoyer les résultats à Paul, sans tenir compte du propriétaire**

**MAILTO=paul**

**#**

**# Chaque jour, 5 minutes après Minuit**

**5 0 \* \* \* $HOME/bin/daily.job >> $HOME/tmp/out 2>&1**

**# Le premier de chaque mois à 14h15 - Résultat envoyé à Paul**

**15 14 1 \* \* $HOME/bin/monthly**

**# Les jours de la semaine à 22h**

**0 22 \* \* 1-5 mail -s "Il est 22h" joe%Joe,%%Où sont tes enfants ?%**

**23 0-23/2 \* \* \* echo "Tous les jours, à 23mn après 0h, 2h, 4h..."**

**5 4 \* \* sun echo "Tous les dimanches à 4h 05"**

**3°) Création d’une crontab par un utilisateur.**

**Chaque utilisateur peut créer son fichier crontab personnel par la commande crontab. Dans ce cas, le fichier créé sera du même nom que l’utilisateur et sera placé dans /var/spool/cron.**

**Pour éditer son fichier crontab, on utilisera la commande** ‘**crontab -e’**. **C’est l’éditeur qui sera défini dans la variable d’environnement EDITOR, qui sera utilisée.**

**Pour supprimer une crontab active, on utilisera la commande** ‘**crontab -r**.

**Pour lister notre table crontab en cours, nous utiliserons la commande** ‘**crontab -**l’

**Pour lister la crontab d’un utilisateur, on utilisera la commande** ‘**crontab -u nom\_user -**l’

**Pour pouvoir utiliser la commande crontab en tant qu’utilisateur normal, il faut plusieurs conditions :**

**Si le fichier cron.allow existe, alors vous devez être mentionnés dans celui-ci pour pouvoir utiliser cette commande. Si le fichier cron.allow n’existe pas, mais que le fichier cron.deny existe, alors vous ne devez pas être mentionné dans celui-ci, si vous désirez utiliser cette commande. Si aucun de ces deux fichiers n’existe, seul le super-utilisateur a le droit d’utiliser cette commande.**

**Attention, si un utilisateur crée une crontab parce qu’il en a le droit et que ce droit lui est retirée, la crontab créée précédemment sera toujours effective** **! Il faudra dans ce cas, supprimer le fichier dans /var/spool/cron.**

**1°) Création d’une tâche.**

**Nous allons créer une tâche /usr/local/bin/menage.sh qui effacera tous les fichiers /tmp/\*.tmp, puis enverra un message dans un fichier /root/jmm indiquant l’heure où s’est réalisée la tâche**.

[root@machine1 bin]#

[root@machine1 bin]# **pwd**

**/usr/local/bin**

[root@machine1 bin]#

[root@machine1 bin]# **more menage.sh**

**#**

**rm /tmp/\*.tmp**

**echo -e "Menage fait à `date`" >> /root/jmm**

**#**

[root@machine1 bin]# **cd /**

[root@machine1 /]# **more /etc/cron.allow**

**/etc/cron.allow: Aucun fichier ou dossier de ce type**

[root@machine1 /]#

[root@machine1 /]# **more /etc/cron.deny**

[root@machine1 /]#

**2°) Lancement de la tâche par le fichier /etc/crontab**

[root@machine1 /]#

[root@machine1 /]# **more /etc/crontab**

**SHELL=/bin/bash**

**PATH=/sbin:/bin:/usr/sbin:/usr/bin**

**MAILTO=root**

**HOME=/**

**# For details see man 4 crontabs**

**# Example of job definition:**

**# .---------------- minute (0 - 59)**

**# | .------------- hour (0 - 23)**

**# | | .---------- day of month (1 - 31)**

**# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...**

**# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat**

**# | | | | |**

**# \* \* \* \* \* user-name command to be executed**

**01 13 \* \* \* root /usr/local/bin/menage.sh**

[root@machine1 /]#

**Vérification de la présence de fichiers .tmp dans le répertoire /tmp**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

**Lancement du processus crond pour que les modifications du fichier /etc/crontab soient prises en compte**

root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 12:55:02 CEST 2014**

[root@machine1 /]#

[root@machine1 /]# **service crond stop**

**Arrêt de crond : [ OK ]**

[root@machine1 /]# **service crond start**

**Démarrage de crond : [ OK** ]

[root@machine1 /]#

[root@machine1 /]# **crontab -l**

**no crontab for root**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 12:59:18 CEST 2014**

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 13:01:04 CEST 2014**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**keyring-KaegP0 lost+found orbit-gdm orbit-root pulse-jQ5g94vN9GFK pulse-mPcoMggWPiYI seahorse-U9N2BP VMwareDnD vmware-root**

[root@machine1 /]#

[root@machine1 /]# **more /root/jmm**

**Menage fait à mer. avril 23 13:01:01 CEST 2014**

[root@machine1 /]#

**Le script menage.sh a bien été lancé à 13H01 car le fichier fichier1.tmp et fichier2.tmp ont été effacés.**

**3°) Lancement du script menage.sh par une tâche dans /etc/cron.d.**

[root@machine1 /]#

[root@machine1 /]# **more /etc/crontab**

**SHELL=/bin/bash**

**PATH=/sbin:/bin:/usr/sbin:/usr/bin**

**MAILTO=root**

**HOME=/**

**# For details see man 4 crontabs**

**# Example of job definition:**

**# .---------------- minute (0 - 59)**

**# | .------------- hour (0 - 23)**

**# | | .---------- day of month (1 - 31)**

**# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...**

**# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat**

**# | | | | |**

**# \* \* \* \* \* user-name command to be executed**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **ls -ls /etc/cron.d**

**total 16**

**4 -rw-r--r--. 1 root root 113 23 nov. 13:43 0hourly**

**4 -rw-r--r--. 1 root root 108 11 oct. 2013 raid-check**

**4 -rw-r--r--. 1 root root 235 22 nov. 22:53 sysstat**

**4 -rw-r--r--. 1 root root 46 23 avril 13:17 tache**

[root@machine1 /]#

[root@machine1 /]# **more /etc/cron.d/tache**

**#**

**30 13 \* \* \* root /usr/local/bin/menage.sh**

**#**

[root@machine1 /]#

root@machine1 /]#

[root@machine1 /]# **touch /tmp/fichier1.tmp /tmp/fichier2.tmp**

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 13:21:57 CEST 2014**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

[root@machine1 /]# **service crond stop**

**Arrêt de crond : [ OK ]**

[root@machine1 /]# **service crond start**

**Démarrage de crond : [ OK ]**

[root@machine1 /]#

[root@machine1 /]# **crontab -l**

**no crontab for root**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 13:27:22 CEST 2014**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

[root@machine1 /]# **date**

mer. avril 23 13:30:07 CEST 2014

[root@machine1 /]# **ls /tmp**

**keyring-KaegP0 lost+found orbit-gdm orbit-root pulse-jQ5g94vN9GFK pulse-mPcoMggWPiYI seahorse-U9N2BP** VMwareDnD vmware-root

[root@machine1 /]#

[root@machine1 /]# **more /root/jmm**

**Menage fait à mer. avril 23 13:01:01 CEST 2014**

**Menage fait à mer. avril 23 13:30:01 CEST 2014**

[root@machine1 /]#

**La tâche /etc/cron.d/tache a été réalisée à 13H30. Le nom de la tâche présente dans /etc/cron.d peut avoir n’importe quel nom.**

**4°) Lancement du script menage.sh par une tâche créée par crontab –e avec root**

**Pas de lancement de menage.sh par /etc/crontab, ni par un script présent dans /etc/cron.d**

[root@machine1 /]#

[root@machine1 /]# **ls /etc/cron.d**

**0hourly raid-check sysstat**

[root@machine1 /]#

[root@machine1 /]# **more /etc/crontab**

**SHELL=/bin/bash**

**PATH=/sbin:/bin:/usr/sbin:/usr/bin**

**MAILTO=root**

**HOME=/**

**# For details see man 4 crontabs**

**# Example of job definition:**

**# .---------------- minute (0 - 59)**

**# | .------------- hour (0 - 23)**

**# | | .---------- day of month (1 - 31)**

**# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...**

**# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat**

**# | | | | |**

**# \* \* \* \* \* user-name command to be executed**

[root@machine1 /]#

**Vérification du contenu dans /var/spool/cron**

[root@machine1 /]#

[root@machine1 /]# **ls /var/spool/cron**

[root@machine1 /]#

[root@machine1 /]#

**Création d’une tâche par la commande crontab -e**

[root@machine1 /]#

[root@machine1 /]# **crontab –e**

**#**

**10 14 \* \* \* /usr/local/bin/menage.sh**

**#**

~

[root@machine1 /]#

**Vérification de la résence d’un fichier portant le nom de l’utilisateur qui a créé la tâche (dans notre exemple root)**

[root@machine1 /]#

[root@machine1 /]# **ls /var/spool/cron**

[root@machine1 /]#

[root@machine1 /]# **crontab -e**

**no crontab for root - using an empty one**

**crontab: installing new crontab**

[root@machine1 /]#

[root@machine1 /]# **ls /var/spool/cron**

**root**

[root@machine1 /]#

[root@machine1 /]# **more /var/spool/cron/root**

**#**

**10 14 \* \* \* /usr/local/bin/menage.sh**

**#**

[root@machine1 /]#

[root@machine1 /]# **touch /tmp/fichier1.tmp /tmp/fichier2.tmp**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

[root@machine1 /]# **service crond stop**

**Arrêt de crond : [ OK ]**

[root@machine1 /]# **service crond start**

**Démarrage de crond : [ OK ]**

[root@machine1 /]#

[root@machine1 /]# **crontab -u root -l**

**#**

**10 14 \* \* \* /usr/local/bin/menage.sh**

**#**

[root@machine1 /]#

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 14:09:12 CEST 2014**

[root@machine1 /]#

[root@machine1 /]# **ls /tmp**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 /]#

[root@machine1 /]# **date**

**mer. avril 23 14:10:08 CEST 2014**

[root@machine1 /]# **ls /tmp**

**keyring-KaegP0 lost+found orbit-gdm orbit-root pulse-jQ5g94vN9GFK pulse-mPcoMggWPiYI seahorse-U9N2BP VMwareDnD vmware-root**

[root@machine1 /]# **more /root/jmm**

**Menage fait à mer. avril 23 13:00:01 CEST 2014**

**Menage fait à mer. avril 23 13:01:01 CEST 2014**

**Menage fait à mer. avril 23 13:30:01 CEST 2014**

**Menage fait à mer. avril 23 14:10:01 CEST 2014**

[root@machine1 /]#

**Attention, les lignes dans le fichier /var/spool/cron/root diffère un peu des autres configurations. Il ne faut pas préciser dans ligne, qui lance la tâche. Les logs sont enregistrés dans /var/log/cron**

**5°) Lancement d’une crontab par un utilisateur normal**

**Créons un utilisateur ‘jmm’ avec les paramètres par défaut. Puis nous verrons les différentes cas que nous pouvons rencontrer.**

[root@machine1 /]#

[root@machine1 /]# **useradd jmm**

[root@machine1 /]#

[root@machine1 /]# **passwd jmm**

**Changement de mot de passe pour l'utilisateur jmm.**

**Nouveau mot de passe** :

**MOT DE PASSE INCORRECT : basé sur un mot du dictionnaire**

**Retapez le nouveau mot de passe :**

**passwd : mise à jour réussie de tous les jetons d'authentification.**

[root@machine1 /]#

**On se connecte avec le user ‘jmm’**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **whoami**

**jmm**

[jmm@machine1 ~]$

**1er cas : /etc/cron.allow présent et vide – le fichier /etc/cron.deny présent et vide**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **crontab -e**

**You (jmm) are not allowed to use this program (crontab)**

**See crontab(1) for more information**

[jmm@machine1 ~]$

**Dans ce cas, un utilisateur normal ne pourra pas lancer de tâche par cron**

**2 ème cas : /etc/cron.allow contient le user jmm et le fichier /etc/cron.deny est absent**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **crontab -e**

**no crontab for jmm - using an empty one**

**crontab: installing new crontab**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **touch /tmp/fichier1.tmp /tmp/fichier2.tmp**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **ls -ls /tmp/fi\***

**0 -rw-rw-r--. 1 jmm jmm 0 25 avril 13:51 /tmp/fichier1.tmp**

**0 -rw-rw-r--. 1 jmm jmm 0 25 avril 13:51 /tmp/fichier2.tmp**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **crontab -l**

**55 13 \* \* \* /usr/local/bin/menage.sh**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **date**

**ven. avril 25 13:55:01 CEST 2014**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **ls /tmp/fi\***

**ls: impossible d'accéder à /tmp/fi\*: Aucun fichier ou dossier de ce type**

[jmm@machine1 ~]$

**Dans ce cas, l’utilisateur présent dans le fichier /etc/cron.allow peut lancer des tâche par cron.**

**3 ème cas les fichier /etc/cron.deny et cron.allow n’existent pas**

[root@machine1 ~]# **su - jmm**

[jmm@machine1 ~]$ **crontab -e**

**You (jmm) are not allowed to use this program (crontab)**

**See crontab(1) for more information**

[jmm@machine1 ~]$

**Quelques commandes avec crontab**

Création d’une tâche

[jmm@machine1 ~]$ **crontab -e**

**crontab: installing new crontab**

Listage des tâches en cours

[jmm@machine1 ~]$ **crontab -l**

**55 14 \* \* \* /usr/local/bin/menage.sh**

Suppression de la tâche

[jmm@machine1 ~]$ **crontab –r**

[jmm@machine1 ~]$

Listage à nouveau

[jmm@machine1 ~]$ **crontab -l**

**no crontab for jmm**

[jmm@machine1 ~]$

**6°) D’autres répertoires qui contiennent des tâches à lancer**

* /etc/cron.daily : exécution quotidienne (chaque jour à 4h02)
* /etc/cron.hourly : exécution chaque heure (chaque heure + 1 minute)
* /etc/cron.weekly : exécution hebdomadaire (le dimanche à 4h22)
* /etc/cron.monthly : exécution mensuelle (le 1er du mois à 4h42)

**6°) Sécurisation de crontab**.

**Mettre les droits 600 aux fichiers /etc/crontab, /etc/cron.deny, /etc/cron.allow.**

**Mettre le droit 700 aux répertoires /etc/cron.d et /etc/cron\* et /var/spool/cron**

**Si on ne veut pas que des utilisateurs, autre que root, utilisent cron, on peut supprimer les fichiers /etc/cron.allow et cron.deny.**

**On peut aussi modifier les droits de /etc/crontab et /etc/cron.d par la commande chattr**

**chattr +i /etc/crontab**

**chattr +i /etc/cron.d**

**chattr +i /etc/cron.daily**

**chattr +i /etc/cron.hourly**

**chattr +i /etc/cron.weekly**

**chattr +i /etc/cron.monthly**

**Cette commande empêchera toutes modifications dans ces repertoire (aucun fichier ajouté, aucun fichier supprimé, aucun fichier modifié)**

**Modification des droits des répertoires et fichiers**

[root@machine1 /]# **chmod 700 /etc/cron.d**

[root@machine1 /]# **chmod 700 /etc/cron.daily**

[root@machine1 /]# **chmod 700 /etc/cron.hourly**

[root@machine1 /]# **chmod 700 /etc/cron.monthly**

[root@machine1 /]# **chmod 700 /etc/cron.weekly**

[root@machine1 /]# **chmod 600 /etc/crontab**

[root@machine1 /]# **chmod 700 /var/spool/cron**

**Suppression des fichiers /etc/cron.allow et /etc/cron.allow**

[root@machine1 etc]# **rm cron.allow cron.deny**

**rm : supprimer fichier « cron.allow » ? y**

**rm: impossible de supprimer « cron.deny »: Aucun fichier ou dossier de ce type**

[root@machine1 etc]#

**Application de la commande chattr sur le fichier /etc/crontab, sur les répertoires /etc/cron\* ainsi que sur le répertoire /var/spool/cron**

[root@machine1 /]#

[root@machine1 /]# **chattr +i /etc/cron.daily**

[root@machine1 /]# **chattr +i /etc/cron.hourly**

[root@machine1 /]# **chattr +i /etc/cron.monthly**

[root@machine1 /]# **chattr +i /etc/cron.weekly**

[root@machine1 /]# **chattr +i /etc/cron.d**

[root@machine1 /]# **chattr +i /etc/crontab**

[root@machine1 /]# **chattr +i /var/spool/cron**

[root@machine1 /]#

**Vérification de notre sécurisation**

[root@machine1 etc]# **rm /etc/crontab**

**rm : supprimer fichier « /etc/crontab » ?** **y**

**rm: impossible de supprimer « /etc/crontab »: Opération non permise**

[root@machine1 etc]#

**INCRON**

***Cron* est un outil qui exécute des commandes à certains moments. *Incron* est un outil créé sur le même modèle que *cron* et qui exécute des commandes en fonction de l'activité sur le système de fichiers. *Incron* permet de vérifier si des modifications, tel que des créations de fichiers, des écritures, des fermetures ou des suppressions ont eu lieu sur un fichier spécifique et dans un répertoire entier**.

Les utilisations possibles d'*incron* sont nombreuses par exemple:

* **recharger la configuration d'un serveur lorsque les fichiers de configuration sont modifié**
* **prévenir un utilisateur de l'arrivée d'une nouveau message**
* **vérifier si des fichiers critiques n'ont pas été modifiés**

**Les différents types d’évènements :**

* **IN\_ACCESS : le fichier a été accédé (lecture) ;**
* **IN\_ATTRIB : les métadonnées du fichier ont été modifiées (permissions, timestamp, attributs étendus, etc) ;**
* **IN\_CLOSE\_WRITE : fermeture d’un fichier avec écriture ;**
* **IN\_CLOSE\_NOWRITE : fermeture d’un fichier sans écriture ;**
* **IN\_CREATE : fichier ou répertoire créé dans un répertoire surveillé ;**
* **IN\_DELETE : fichier ou répertoire supprimé dans un répertoire surveillé ;**
* **IN\_DELETE\_SELF : le fichier ou répertoire surveillé est supprimé ;**
* **IN\_MODIFY : le fichier ou répertoire a été modifié ;**
* **IN\_MOVE\_SELF : le fichier ou répertoire a été déplacé ;**
* **IN\_MOVED\_FROM : un fichier du répertoire surveillé à été déplacé ;**
* **IN\_MOVED\_TO : un fichier à été déplacé dans le répertoire surveillé ;**
* **IN\_OPEN : le fichier à été ouvert ;**
* **IN\_ALL\_EVENTS : combinaison de tous ce qui précède.**
* **IN\_MOVE : équivalent à IN\_MOVED\_FROM ou IN\_MOVED\_TO ;**
* **IN\_CLOSE : équivalent à IN\_CLOSE\_WRITE ou IN\_CLOSE\_NOWRITE ;**

**Plusieurs valeurs sont possibles séparées par des virgules.**

**ATTENTION : incron n’est pas récursif ! Seules les modifications intervenues au premier niveau du répertoire surveillé seront détectées.**

**La commande peut utiliser un certain nombre de variables prédéfinies :**

* **$@ : chemin complet du fichier/répertoire surveillé ;**
* **$# : nom du fichier/répertoire sur lequel l’événement est intervenu ;**
* **$% : nom (textuel) de l’événement intervenu ;**
* **$& : numéro de l’événement intervenu.**
* **$$ : le caractère dollar (“$”) ;**

**Fichiers de configuration de incrond**

**Dans le répertoire /etc se trouvent 3 fichiers de configuration :  
  
incron.conf : fichier de configuration principal  
incron.allow : utilisateurs autorisés à utiliser incron  
incron.deny : utilisateurs non autorisés à utiliser incron  
  
Si le fichier incron.allow existe, seuls les utilisateurs listés pourront utiliser incron. Même pricipe qu’avec le processus cron.**

**Le fichier /etc/incron.conf**

[root@machine1 /]# cd /etc

[root@machine1 etc]# more incron.conf

#

# \*\*\* incron example configuration file \*\*\*

#

# (c) Lukas Jelinek, 2007, 2008

#

# Parameter: system\_table\_dir

# Meaning: system table directory

# Description: This directory is examined by incrond for system table files.

# Default: /etc/incron.d

#

# Example:

# system\_table\_dir = /var/spool/incron.systables

# Parameter: user\_table\_dir

# Meaning: user table directory

# Description: This directory is examined by incrond for user table files.

# Default: /var/spool/incron

#

# Example:

# user\_table\_dir = /var/spool/incron.usertables

# Parameter: allowed\_users

# Meaning: allowed users list file

# Description: This file contains users allowed to use incron.

# Default: /etc/incron.allow

#

# Example:

# allowed\_users = /etc/incron/allow

# Parameter: denied\_users

# Meaning: denied users list file

# Description: This file contains users denied to use incron.

# Default: /etc/incron.deny

#

# Example:

# denied\_users = /etc/incron/deny

# Parameter: lockfile\_dir

# Meaning: application lock file directory

# Description: This directory is used for creating a lock avoiding to run

# multiple instances of incrond.

# Default: /var/run

#

# Example:

# lockfile\_dir = /tmp

# Parameter: lockfile\_name

# Meaning: application lock file name base

# Description: This name (appended by '.pid') is used for creating a lock

# avoiding to run multiple instances of incrond.

# Default: incrond

#

# Example:

# lockfile\_name = incron.lock

# Parameter: editor

# Meaning: editor executable

# Description: This name or path is used to run as an editor for editting

# user tables.

# Default: vim

#

# Example:

# editor = nano

[root@machine1 etc]#

**Exemple** :

**1°) Nous allons relancer le service vsftpd chaque fois que le fichier /etc/vsftpd/vsftpd.conf sera modifié**

**Lancement du service incrond**

[root@machine1 /]#

[root@machine1 /]# **service incrond start**

**Démarrage de Filesystem event daemon (incrond) : [ OK ]**

[root@machine1 /]#

**Création d’un fichier incrond avec la commande incrontab -e**

[root@machine1 11 ème Atelier INCROND]#

[root@machine1 11 ème Atelier INCROND]# **incrontab -e**

[root@machine1 11 ème Atelier INCROND]#

**Cette commande va créer un fichier dans le répertoire /var/spool/incron/root (root étant l’utilisateur qui aura crée une ‘incrontab’)**

[root@machine1 incron]#

[root@machine1 incron]# **pwd**

**/var/spool/incron**

[root@machine1 incron]# **ls -ls**

**total 4**

**4 -rw-------. 1 root root 62 10 juin 10:58 root**

[root@machine1 incron]# **more root**

**/etc/vsftpd/vsftpd.conf IN\_CLOSE\_WRITE service vsftpd restart**

[root@machine1 incron]#

**Description des champs des lignes du fichier de commande**

**/etc/vsftpd/vsftpd.conf : fichier à surveiller**

**IN\_CLOSE\_WRITE : exécuter la commande lorsque le fichier est modifié et fermé**

**Service vsftpd restart : commande à exécuter. Dans notre cas, lancement du service vsftpd.**

**On peut vérifier les tâches prises en compte par le processus incrond**

[root@machine1 /]#

[root@machine1 /]# **incrontab -l**

**/etc/vsftpd/vsftpd.conf IN\_CLOSE\_WRITE service vsftpd restart**

[root@machine1 /]#

**Pour contrôler que le processus se lance bien au moment de la modification du fichier /etc/vsftpd/vsftpd.conf,**

**Lancer dans une fenêtre la commande suivante tail –f /var/log/cron et dans l’autre touch /etc/vsftpd/vsftpd.conf**

**2°) Surveillance d’un ou plusieurs répertoires.**

Voici notre nouveau incrontab

[root@machine1 /]#

[root@machine1 /]# **more /var/spool/incron/root**

**/etc/vsftpd/vsftpd.conf IN\_CLOSE\_WRITE service vsftpd restart**

**/etc/ IN\_CLOSE\_WRITE /usr/local/bin/message.sh $@/$# fichier**

**/var/spool/cron IN\_ALL\_EVENTS /usr/local/bin/message.sh $@ reperoire**

[root@machine1 /]#

La première ligne indique que le service vsftpd sera relancé à chaque modification du ficher /etc/vsftpd/vsftpd.conf

La deuxième ligne indique que le script /usr/local/bin/message.sh sera exécuté à chaque modification d’un dichier de configuration dans /etc

La troisième ligne indique que le script /usr/local/bin/message.sh sera lancé pour tous les évènements dans le répertoire /var/spool/cron

Voici le script /usr/local/bin/message.sh

[root@machine1 mail]#

root@machine1 mail]# **more /usr/local/bin/message.sh**

**echo -e "Attention le $2 $1 a ete modifie" | mail -s "modification fichier" user01@localhost**

[root@machine1 mail]#

Le script message.sh envoie un mail à l’utilisateur user01 en lui indiquant le répertoire ou fichier modifié. Le nom de ce répertoire ou fichier est passé en paramètre de position.

[root@machine1 mail]#

[root@machine1 mail]# **pwd**

**/var/spool/mail**

[root@machine1 mail]# **ls**

**root rpc**

[root@machine1 mail]#

On remarque que l’utilisateur n’a pas de courrier

**Démarrage du service incrond**

[root@machine1 incron]# **service incrond stop**

**Arrêt de Filesystem event daemon (incrond) : [ OK ]**

[root@machine1 incron]# **service incrond start**

**Démarrage de Filesystem event daemon (incrond) : [ OK ]**

[root@machine1 incron]#

**Vérification de l’heure pour les logs**

[root@machine1 incron]# **date**

**lun. juin 16 12:29:25 CEST 2014**

[root@machine1 incron]# **touch /etc/vsftpd/vsftpd.conf**

**Modification de /etc/hosts.conf pour la deuxième directive et la creation d’un repertoire dans /var/spool/cron pour la troisième directive**

[root@machine1 incron]# **touch /etc/hosts.conf**

[root@machine1 incron]#

[root@machine1 incron]# **mkdir /var/spool/cron/ROOT**

[root@machine1 incron]#

Vérification dans les logs

[root@machine1 incron]# **tail -8 /var/log/cron**

**Jun 16 12:29:21 machine1 incrond[4274]: starting service (version 0.5.9, built on Mar 10 2011 08:55:04)**

**Jun 16 12:29:21 machine1 incrond[4275]: loading system tables**

**Jun 16 12:29:21 machine1 incrond[4275]: loading user tables**

**Jun 16 12:29:21 machine1 incrond[4275]: loading table for user root**

**Jun 16 12:29:21 machine1 incrond[4275]: ready to process filesystem events**

**Jun 16 12:29:47 machine1 incrond[4275]: (root) CMD (service vsftpd restart)**

**Jun 16 12:30:00 machine1 incrond[4275]: (root) CMD (/usr/local/bin/message.sh /etc//hosts.conf fichier)**

**Jun 16 12:30:33 machine1 incrond[4275]: (root) CMD (/usr/local/bin/message.sh /var/spool/cron repertoire)**

[root@machine1 incron]#

Vérification de la messagerie de user01

[root@machine1 incron]# **more /var/spool/mail/user01**

From root@machine1.stage.fr Mon Jun 16 12:30:01 2014

Return-Path: <root@machine1.stage.fr>

X-Original-To: user01@localhost

Delivered-To: user01@localhost.stage.fr

Received: by machine1.stage.fr (Postfix, from userid 0)

id F09C210AD; Mon, 16 Jun 2014 12:30:00 +0200 (CEST)

Date: Mon, 16 Jun 2014 12:30:00 +0200

To: user01@localhost.stage.fr

Subject: modification fichier

User-Agent: Heirloom mailx 12.4 7/29/08

MIME-Version: 1.0

Content-Type: text/plain; charset=us-ascii

Content-Transfer-Encoding: 7bit

Message-Id: <20140616103000.F09C210AD@machine1.stage.fr>

From: root@machine1.stage.fr (root)

**Attention le fichier /etc//hosts.conf a ete modifie**

**From root@machine1.stage.fr Mon Jun 16 12:30:33 2014**

Return-Path: <root@machine1.stage.fr>

X-Original-To: user01@localhost

Delivered-To: user01@localhost.stage.fr

Received: by machine1.stage.fr (Postfix, from userid 0)

id 64B1B10B0; Mon, 16 Jun 2014 12:30:33 +0200 (CEST)

Date: Mon, 16 Jun 2014 12:30:33 +0200

To: user01@localhost.stage.fr

Subject: modification fichier

User-Agent: Heirloom mailx 12.4 7/29/08

MIME-Version: 1.0

Content-Type: text/plain; charset=us-ascii

Content-Transfer-Encoding: 7bit

Message-Id: <20140616103033.64B1B10B0@machine1.stage.fr>

From: root@machine1.stage.fr (root)

**Attention le repertoire /var/spool/cron a ete modifie**

[root@machine1 incron]#

**AT**

**at et batch lisent, depuis l’entrée standard, ou depuis un fichier, des commandes qu’ils exécuteront ultérieurement, en utilisant**

**/bin/sh.**

**at lance les commandes indiquées à une heure précise.**

**atq affiche la liste des commandes en attente d’exécution pour l’utilisateur appelant, sauf s’il s’agit du super-utilisateur**

**auquel cas toutes les commandes en attente sont listées. Le format de sortie est (une ligne par job) : numéro de job, date,**

**heure, file et nom d’utilisateur.**

**atrm efface les travaux en attente, identifiés par leurs numéros de job.**

**batch exécute les commandes indiquées lorsque la charge système le permet, c’est-à-dire lorsque la charge du processeur descend sous 0.8 (Ndt : cette valeur peut être différente suivant les distributions car elle est configurable à la construction du**

**binaire), ou en dessous d’une valeur mentionnée explicitement durant l’invocation de atd.**

**at permet d’indiquer l’heure de lancement de manière assez complexe, en extension de la norme POSIX.2. Il accepte des spécifications de la forme HH:MM pour exécuter un travail à une heure donnée de la journée en cours (ou du lendemain si l’heure mentionnée est déjà dépassée). On peut aussi lui fournir l’un des arguments suivants : midnight (minuit), noon (midi), ou teatime (l’heure du thé, soit 16 heures). Il est également possible de fournir un suffixe du type AM (matin), ou PM (après-midi) avec une heure sur un cadran de 12 heures. On peut indiquer le jour de lancement, en précisant une date du type nom-du-mois (en anglais) jour et éventuellement année, ou encore une date du type MMJJAA ou MM/JJ/AA ou JJ.MM.AA. La date doit suivre l’heure. On peut indiquer une heure de la forme now (maintenant) + nombre d’unités, où les unités peuvent être minutes, hours (heures), days (jours), ou weeks (semaines) et on peut également demander à at de déclencher le travail le jour même en ajoutant le suffixe today ou le lendemain en ajoutant tomorrow.**

**Par exemple, pour lancer une commande à 16 heures, 3 jours plus tard, on peut indiquer at 4pm + 3 days, pour lancer le travail à 10 heures du matin le 31 juillet, il faut invoquer at 10am Jul 31 et pour lancer une commande à 1 heure du matin le lendemain, effectuez at 1am tomorrow.**

**Aussi bien at que batch lisent les commandes à mémoriser depuis l’entrée standard, ou dans le fichier indiqué par l’option -f. Le**

**répertoire de travail, l’environnement (sauf pour les variables TERM, DISPLAY et \_) ainsi que le umask sont mémorisés au moment de l’invocation. L’appelant recevra par courriel (mail) les messages écrits sur les flux de sortie standard et d’erreur standard, s’il y en a.**

**Le super-utilisateur peut toujours employer ces commandes. Pour les autres utilisateurs, les permissions sont déterminées par les fichiers /etc/at.allow et /etc/at.deny. Si le fichier /etc/at.allow existe, seuls les utilisateurs dont les noms sont mentionnés dans ce fichier peuvent utiliser at.**

**Si /etc/at.allow n’existe pas, at vérifie si /etc/at.deny existe, et tous les utilisateurs non-mentionnés dans ce fichier ont le droit**

**d’invoquer at.**

**Si aucun de ces deux fichiers n’existe, seul le super-utilisateur a le droit d’appeler at.**

**Un fichier /etc/at.deny vide signifie que tous les utilisateurs ont le droit d’appeler ces commandes, c’est la configuration par**

**défaut.**

**OPTIONS**

**-V Afficher un numéro de version sur la sortie d’erreur standard.**

**-q file Utiliser la file d’attente mentionnée. Une file est désignée par une lettre unique, dans l’intervalle a jusqu’à z, et A**

**jusqu’à Z. La file a est la file d’attente par défaut pour at tandis que la file b est celle par défaut pour batch. Plus les**

**files ont une lettre importante, plus les travaux seront exécutés avec une valeur de gentillesse (voir nice(1)) élevée. La**

**file spéciale « = » est réservée pour les jobs en cours d’exécution.**

**Si une tâche est soumise à une file désignée par une lettre majuscule, celle-ci est traitée par batch à l’heure prévue. Une fois**

**l’horaire atteint, les règles de traitement par lots s’appliquent selon de la charge moyenne. Si l’on invoque atq avec un nom de file**

**spécifique, il ne montrera que les travaux en attente dans cette file.**

**-m Envoyer un courrier à l’utilisateur lorsque le travail est terminé, même s’il n’a rien écrit sur ses flux de sortie.**

**-f fichier Lire la commande à exécuter dans le fichier et non pas sur l’entrée standard.**

**-l Est un alias pour atq.**

**-d Est un alias pour atrm.**

**-v Afficher les heures de lancement programmées avant de lire le job. Les heures seront affichées avec le format « Thu Feb 20 14:50:00 1997 ».**

**-c Imprimer sur la sortie standard les travaux mentionnés sur la ligne de commande.**

**Nous programmons la tâche /usr/local/bin/menage.sh à 15 :25 le 23 Avril 2014**

[root@machine1 tmp]# **date**

**mer. avril 23 15:22:12 CEST 2014**

[root@machine1 tmp]#

[root@machine1 tmp]# **ls**

**fichier1.tmp keyring-KaegP0 orbit-gdm pulse-jQ5g94vN9GFK seahorse-U9N2BP vmware-root**

**fichier2.tmp lost+found orbit-root pulse-mPcoMggWPiYI VMwareDnD**

[root@machine1 tmp]#

root@machine1 tmp]# **at -f /usr/local/bin/menage.sh 15:25 042314**

**job 1 at 2014-04-23 15:25**

[root@machine1 tmp]# **atq**

**1 2014-04-23 15:25 a root**

[root@machine1 tmp]#

[root@machine1 tmp]# **date**

**mer. avril 23 15:25:01 CEST 2014**

[root@machine1 tmp]# **ls**

**keyring-KaegP0 lost+found orbit-gdm orbit-root pulse-jQ5g94vN9GFK pulse-mPcoMggWPiYI seahorse-U9N2BP** **VMwareDnD vmware-root**

[root@machine1 tmp]# **atq**

[root@machine1 tmp]#

[root@machine1 tmp]# **more /root/jmm**

**Menage fait à mer. avril 23 13:00:01 CEST 2014**

**Menage fait à mer. avril 23 13:01:01 CEST 2014**

**Menage fait à mer. avril 23 13:30:01 CEST 2014**

**Menage fait à mer. avril 23 14:10:01 CEST 2014**

**Menage fait à mer. avril 23 15:25:00 CEST 2014**

[root@machine1 tmp]#

**SUDO**

**1 Qu’est que c’est ? Pourquoi utiliser sudo ?**

**Par défaut un utilisateur sur un système Linux, n’a que très peu de droits sur le système, il possède les droits de lecture et écriture dans son $HOME, et ne peut (généralement) lancer que les commandes destinées aux utilisateurs (celles qui se trouvent dans /bin, /usr/bin, /usr/local/bin).**

**Il peut arriver qu’un utilisateur normal ait besoin de lancer certaines commandes avec les droits root ou les droits d’un autre utilisateur, sudo permet de faire ceci et ce sans connaitre le mot de passe de cet utilisateur (ce qui est un avantage notable par rapport à la commande su).**

**Sudo est donc un petit programme permettant à l’administrateur d’une machine de donner des droits supplémentaires à des utilisateurs normaux, il permet aussi de « tracer » tout ce qui est fait par les utilisateurs (nécessaire dans certains environnements sécurisés).**

* 1. **Syntaxe**

|  |  |
| --- | --- |
| 1  2 | ## Allow root to run any commands anywhere – Ligne commentaire  root    ALL=(ALL)       ALL |

Explications :

*root* : Utilisateur à qui on donne des droits

*ALL* : (n’importe quelle machine)

*(ALL)* : (en tant que n’importe qui)

*ALL* : ( pour toutes les commandes)

**2 Configuration**

**La configuration de sudo se fait au travers du fichier /etc/sudoers.**

|  |  |
| --- | --- |
|  |  |

**Ce fichier ne doit pas être édité avec un éditeur classique, mais avec la commande visudo.**

**visudo (qui lancera par défaut vi) effectue quelques tâches de vérification quant à la syntaxe du fichier, vous évitant ainsi des erreurs.**

**L’objectif est de définir des groupes de machines, des groupes de commandes, et des groupes d’utilisateurs, pour ensuite associer les uns aux autres.**

**Le fichier de configuration est composé de deux types d’entrées :**

* **Les alias.**
* **Les spécifications utilisateurs**.

**2.1 Les alias**

Les alias sont généralement des variables, représentant quatre types d’objets : les utilisateurs, les exécutants, les hôtes et les commandes.

**2.1.1 User\_Alias : les utilisateurs**

Cet alias permet de définir un groupe d’utilisateurs, ainsi pour définir un groupe nommé *NETWORK-TEAM* comprenant les utilisateurs *alfred*, *brenda* et *charly*, il faudra utiliser la syntaxe suivante :

User\_Alias NETWORK-TEAM = alfred, brenda, charly

**2.1.2 Runas\_Alias : les exécutants**

Les alias Runas\_Alias permettent d’indiquer en tant que quel utilisateur ( ou groupe ) sera lancé une commande.  
Ainsi, pour indiquer que l’utilisateur *alfred* peut lancer en tant que n’importe quel utilisateur du Runas\_Alias DB (oracle, sybase) n’importe quelle commande sans donner de mot de passe, il faudra ajouter les lignes suivantes dans le fichier /etc/sudoers :

Runas\_Alias DB = oracle, sybase

alfred ALL = (DB) NOPASSWD: ALL

De cette façon, pour lancer une commande en tant que oracle ou sybase, l’utilisateur alfred devra lancer par exemple :

[alfred@server ~]$ sudo -u oracle /sbin/oracle restart

**2.1.3 Hosts\_Alias : les hôtes**

Cet alias quand à lui indique une liste de machines sur lesquelles vous pourrez utiliser ou pas certaines commandes, dans le cas où le fichier est identique sur plusieurs machines.

La syntaxe suivante défini trois ensembles de machines :

Host\_Alias FILESERVERS = fs1, fs2

Host\_Alias DEVSERVER = dev1, dev2

Host\_Alias DBSERVER = db1, db2

**2.1.4 Cmnd\_Alias : les commandes**

Cet alias permet de grouper plusieurs commandes sous un seul nom, c’est ce nom qui sera utilisé pour la délégation des droits, les lignes suivantes permettent de regrouper plusieurs commandes en rapport avec le réseau nommée *NETWORKING* :

Cmnd\_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net, /sbin/iptables, /sbin/iwconfig, /sbin/mii-tool, /usr/sbin/mtr, /sbin/ip

**2.2 Les spécifications utilisateurs**

La dernière partie de la configuration de sudo, est l’attribution des différents alias de commande aux différents alias utilisateurs. De cette façon, en utilisant les exemples précédents, nous pouvons attribuer les commandes en relation avec le réseau aux utilisateurs *alfred*, *charly* et *brenda* :

User\_Alias NETWORK-TEAM = alfred, brenda, charly

Cmnd\_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net, /sbin/iptables, /sbin/iwconfig, /sbin/mii-tool, /usr/sbin/mtr, /sbin/ip

NETWORK-TEAM ALL = NETWORKING

Il est donc aisé d’ajouter un peu plus de complexité au fichier /etc/sudoers. Imaginons que les utilisateurs ne puissent agir sur le réseau que depuis certaines machines (ici le fichier/etc/sudoers est commun à plusieurs machines ) :

# Définitions des machines :

Host\_Alias FILESERVERS = fs1, fs2

Host\_Alias DEVSERVER = dev1, dev2

Host\_Alias DBSERVER = db1, db2

# Définition des utilisateurs :

User\_Alias NETWORK-TEAM = alfred, brenda, charly

# Définition des Runas\_Alias

Runas\_Alias DB = oracle, sybase

# Définition des "jeux" de commandes :

Cmnd\_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net, /sbin/iptables, /sbin/iwconfig, /sbin/mii-tool, /usr/sbin/mtr, /sbin/ip

Cmnd\_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum

# Spécifications :

NETWORK-TEAM FILESERVERS = NETWORKING

NETWORK-TEAM DEVSERVER = SOFTWARE

NETWORK-TEAM DBSERVER = (DB) NOPASSWD: ALL

L’exemple ci-dessus indique que sur les machines *fs1* et *fs2* les utilisateurs *alfred*, *brenda* et *charly* pourront utiliser les commandes de l’alias *NETWORKING*. Par contre sur les machines *dev1* et *dev2* ils ne pourront utiliser que les commandes de l’alias *SOFTWARE*.

**3 Utilisation**

L’utilisation de **sudo** est très simple, il suffit de lancer la commande souhaitée précédée de sudo. Pour lister les commandes autorisées il suffit de lancer :

[alfred@server ~]$ sudo -l

[sudo] password for alfred :

User alfred may run the following commands on this host:

(root) /sbin/route, /sbin/ifconfig, /bin/ping,

/sbin/dhclient, /usr/bin/net, /sbin/iptables, /usr/bin/rfcomm,

/usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool, (root) /bin/rpm,

/usr/bin/up2date, /usr/bin/yum, (root) /sbin/service, /sbin/chkconfig, (root)

/sbin/fdisk, /sbin/sfdisk, /sbin/parted, /sbin/partprobe, /bin/mount,

/bin/umount, (root) /usr/sbin/visudo, /bin/chown, /bin/chmod, /bin/chgrp,

(root) /bin/nice, /bin/kill, /usr/bin/kill, /usr/bin/killall, (root)

/usr/bin/updatedb, (root) /sbin/modprobe

Et pour utiliser l’une de ces commandes :

[alfred@server ~]$ sudo /sbin/chkconfig httpd on

Et dans le cas d’utilisation de Runas\_Alias :

[alfred@server ~]$ sudo -u oracle /sbin/oracle restart

**4 Conclusion**

La configuration de sudo est parfois un peu laborieuse, mais c’est le prix à payer pour une administration sereine d’un système. La page de man de sudoers (man 5 sudoers) comporte de nombreux exemples d’utilisation avancée de sudo. Il faut faire aussi attention à ne pas se laisser tenter pas une délégation de droits trop simpliste comme :

admin ALL = (ALL) NOPASSWD: ALL

Autant se logger en root !!

EXEMPLE **Nous allons donner le droit à l’utilisateur ‘jmm’ d’arrêter le système sans connaitre le mot de passe de root.**

**Modification du fichier /etc/sudoers**

[root@machine1 etc]#

[root@machine1 etc]# **tail sudoers**

**## Allows members of the users group to mount and unmount the**

**## cdrom as root**

**# %users ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom**

**## Allows members of the users group to shutdown this system**

**# %users localhost=/sbin/shutdown -h now**

**## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)**

**#includedir /etc/sudoers.d**

**jmm machine1= NOPASSWD: /sbin/shutdown -h**

**[root@machine1 etc]#**

**Ici, on va permettre à l’utilisateur ‘jmm’ d’arrêter l’ordinateur sans connaître le mot de passe.**

**Sudo n’est pas dépendant d’un service. Toute modification du fichier /etc/sudoers est directement prise en compte**.

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **whoami**

**jmm**

[jmm@machine1 ~]$ **sudo -l**

**Matching Defaults entries for jmm on this host:**

**requiretty, !visiblepw, always\_set\_home, env\_reset, env\_keep="COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS\_COLORS", env\_keep+="MAIL PS1 PS2 QTDIR**

**USERNAME LANG LC\_ADDRESS LC\_CTYPE", env\_keep+="LC\_COLLATE LC\_IDENTIFICATION LC\_MEASUREMENT LC\_MESSAGES", env\_keep+="LC\_MONETARY LC\_NAME LC\_NUMERIC**

**LC\_PAPER LC\_TELEPHONE", env\_keep+="LC\_TIME LC\_ALL LANGUAGE LINGUAS \_XKB\_CHARSET XAUTHORITY", secure\_path=/sbin\:/bin\:/usr/sbin\:/usr/bin**

**User jmm may run the following commands on this host:**

**(root) NOPASSWD: /sbin/shutdown -h**

[jmm@machine1 ~]$

[jmm@machine1 ~]$ **sudo -u root /sbin/shutdown -h now**

**PROCESSUS LANCES AU DEMARRAGE**

1. **Le fichier /etc/inittab**

Les daemons (ou démons) sont des programmes résidents chargés au démarrage. A chaque runlevel, correspond une liste de daemons à lancer (1 à 5) ou à arrêter (6 ou 0).

D'autres programmes que des démons peuvent également être lancés dès le démarrage de la machine, avec le même mécanisme.

Les runlevels ou "niveaux d'exécution", correspondent aux services qui vont être lancés au démarrage de la machine. En général (mais toutes les distributions n'utilisent pas la même numérotation), on peut avoir les niveaux d'éxécution suivant :

**- le niveau 0 (halt) est celui dans lequel doit passer le système lorsqu'on désire arrêter la machine, afin de le faire "proprement", notamment en vérifiant que toutes les informations qui doivent être écrites sur le disque , l'ont effectivement été.**

**- le niveau 1 est un mode mono-utilisateur : seul root peut accéder à la console. Ce mode n'est normalement utilisé que pour des opérations de maintenance sensibles, qui nécessitent un fonctionnement minimal du système pour éviter les interférences dues à d'autres utilisateurs. Pratiquement aucun script n'est lancé.**

**- le niveau 2 est un mode multi-utilisateurs, sans support réseau. On retrouve les six consoles texte virtuelles usuelles. Les scripts spécifiques à la gestion du réseau ne sont pas exécutés.**

**- le niveau 3 est identique au niveau 2, mais avec le support réseau. C'est le mode de fonctionnement usuel.**

**- le niveau 4 est par défaut identique au niveau 3, mais on peut l'utiliser pour ses propres configurations personnalisées.**

**- le niveau 5 identique au niveau 3 mais en plus démarre en mode graphique.**

**- le niveau 6 (reboot) est identique au niveau 0, mais en plus il provoque un redémarrage à chaud de l'ordinateur (c'est un équivalent de la combinaison Alt-Ctrl+Suppr)**

Le niveau d'exécution est déterminé (dans l'ordre) soit :

* lors du boot : si vous précisez un niveau sur la ligne de commande du noyau (par exemple, au prompt LILO, taper "linux 1"),
* dans le fichier /etc/inittab, où le runlevel par défaut est défini,
* par la commande init <runlevel> qui permet de changer de runlevel en cours de fonctionnement.

**Le fichier /etc/inittab décrit l’ensemble des processus qui doivent être lancés au démarrage du système. Typiquement, une entrée dans le fichier /etc/inittab a la forme suivante** **:**

**id:runlevels:action:process**

**Les lignes commençant par le caractère « # » sont ignorées.**

**id Séquence unique de 1 à 4 caractères identifiant une entrée dans inittab.**

**Note : pour les programmes de connexion comme les gettys ou d’autres, le champ id doit être le numéro du tty correspondant à la console, par exemple 1 pour tty1. Dans le cas contraire, la connexion risquerait de ne pas bien fonctionner.**

**runlevels**

**Liste des modes de démarrage pour lesquels l’action doit être faite.**

**action**

**Décrit l’action à faire.**

**process**

**Spécifie la commande à exécuter. Si ce champ commence par le caractère « + », init ne lancera pas les commandes utmp et wtmp pour enregistrer les connexions. Ceci est requis par getty qui utilise son propre gestionnaire d’enregistrement de connexion.**

[root@machine1 ~]#

[root@machine1 ~]# **more /etc/inittab**

**# inittab is only used by upstart for the default runlevel.**

**#**

**# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.**

**#**

**# System initialization is started by /etc/init/rcS.conf**

**#**

**# Individual runlevels are started by /etc/init/rc.conf**

**#**

**# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf**

**#**

**# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/serial.conf,**

**# with configuration in /etc/sysconfig/init.**

**#**

**# For information on how to write upstart event handlers, or how**

**# upstart works, see init(5), init(8), and initctl(8).**

**#**

**# Default runlevel. The runlevels used are:**

**# 0 - halt (Do NOT set initdefault to this)**

**# 1 - Single user mode**

**# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)**

**# 3 - Full multiuser mode**

**# 4 - unused**

**# 5 - X11**

**# 6 - reboot (Do NOT set initdefault to this)**

**#**

**id:5:initdefault:**

[root@machine1 ~]#

1. **Fonctionnement.**

**Les services sont lancés par des scripts situés dans /etc/init. Chaque script contient une description ce qui permet de savoir ce que fait chaque daemon en début de script.**

**Le répertoire /etc/rc.d/ contient aussi des répertoires nommés rcX.d (avec X numéro de runlevel). Chacun de ces répertoires contient un lien vers les scripts situés dans init.d.**

**Extrait du répertoire /etc/init.d**

[root@machine1 init.d]# **ls -ls | more**

**total 552**

**4 -rwxr-xr-x. 1 root root 1288 22 nov. 2013 abrt-ccpp**

**4 -rwxr-xr-x. 1 root root 1628 22 nov. 2013 abrtd**

**4 -rwxr-xr-x. 1 root root 1642 22 nov. 2013 abrt-oops**

**4 -rwxr-xr-x. 1 root root 1725 18 août 2010 acpid**

**…**

**Extrait de /etc/rc.d/rcx.d**

[root@machine1 init.d]# **cd /etc/rc.d/rc5.d**

[root@machine1 rc5.d]# **ls -ls | more**

**total 0**

**0 lrwxrwxrwx. 1 root root 15 2 juin 15:24 K01numad -> ../init.d/numad**

**0 lrwxrwxrwx. 1 root root 16 2 juin 15:24 K01smartd -> ../init.d/smartd**

**0 lrwxrwxrwx. 1 root root 17 2 juin 15:24 K02oddjobd -> ../init.d/oddjobd**

**0 lrwxrwxrwx. 1 root root 16 2 juin 15:24 K05conman -> ../init.d/conman**

**0 lrwxrwxrwx. 1 root root 17 2 juin 15:24 K100mcelogd -> ../init.d/mcelogd**

**…**

**0 lrwxrwxrwx. 1 root root 15 2 juin 15:24 S90crond -> ../init.d/crond**

**0 lrwxrwxrwx. 1 root root 13 2 juin 15:24 S95atd -> ../init.d/atd**

**0 lrwxrwxrwx. 1 root root 20 2 juin 15:24 S99certmonger -> ../init.d/certmonger**

**0 lrwxrwxrwx. 1 root root 24 2 juin 15:24 S99libvirt-guests -> ../init.d/libvirt-guests**

**0 lrwxrwxrwx. 1 root root 11 17 janv. 15:31 S99local -> ../rc.local**

[root@machine1 rc5.d]#

**La 1ère lettre détermine si le daemon est activé (S comme start) dans ce niveau d'exécution (runlevel) ou arrêté (K comme kill).   
Les 2 chiffres permettent de trier l'ordre d'exécution des services**

**La commande chkconfig**

**Pour configurer un lancement automatique de processus au démarrage, on indique le niveau qui est associé à ce processus.**

**[root@machine1 ~]# chkconfig --list**

**NetworkManager 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**abrt-ccpp 0:arrêt 1:arrêt 2:arrêt 3:marche 4:arrêt 5:marche 6:arrêt**

**abrtd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:arrêt 5:marche 6:arrêt**

**acpid 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**atd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**auditd 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**autofs 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**bgpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**blk-availability 0:arrêt 1:marche 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**certmonger 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**cgconfig 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**cgred 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**conman 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**cpuspeed 0:arrêt 1:marche 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**crond 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**cups 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**dhcpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**dhcpd6 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**dhcrelay 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**dirsrv 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**dirsrv-snmp 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**dnsmasq 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**firstboot 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**haldaemon 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**htcacheclean 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**httpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ip6tables 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**ipa 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**iptables 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**irqbalance 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**kadmin 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**kdump 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**kprop 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**krb5kdc 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**libvirt-guests 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**lvm2-monitor 0:arrêt 1:marche 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**mcelogd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:arrêt 5:arrêt 6:arrêt**

**mdmonitor 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**memcached 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**messagebus 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**mysqld 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**named 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**netconsole 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**netfs 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**network 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**nfs 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**nfslock 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**nmb 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**nscd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**nslcd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ntpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ntpdate 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**numad 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**oddjobd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**openct 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**ospf6d 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ospfd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**pcscd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**pki-cad 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**portreserve 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**postfix 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**psacct 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**quota\_nld 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**radiusd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**radvd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**rdisc 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**restorecond 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ripd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ripngd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**rngd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**rpcbind 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**rpcgssd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**rpcsvcgssd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**rsyslog 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**saslauthd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**slapd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**smartd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**smb 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**snmpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**snmptrapd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**sshd 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**sssd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**stap-server 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**svnserve 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**sysstat 0:arrêt 1:marche 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**tomcat6 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**udev-post 0:arrêt 1:marche 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**uuidd 0:arrêt 1:arrêt 2:marche 3:marche 4:marche 5:marche 6:arrêt**

**vsftpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**winbind 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**wpa\_supplicant 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**xinetd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:marche 5:marche 6:arrêt**

**ypbind 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**yppasswdd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ypserv 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**ypxfrd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**zebra 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

**services basés sur xinetd :**

**chargen-dgram: arrêt**

**chargen-stream: arrêt**

**cvs: arrêt**

**daytime-dgram: arrêt**

**daytime-stream: arrêt**

**discard-dgram: arrêt**

**discard-stream: arrêt**

**echo-dgram: arrêt**

**echo-stream: arrêt**

**eklogin: arrêt**

**ekrb5-telnet: arrêt**

**gssftp: arrêt**

**klogin: arrêt**

**krb5-telnet: arrêt**

**kshell: arrêt**

**rsync: arrêt**

**tcpmux-server: arrêt**

**tftp: arrêt**

**time-dgram: arrêt**

**time-stream: arrêt**

**[root@machine1 ~]#**

**La commande chkconfig –list nous permet de connaitre les processus et leur niveau associé. Nous allons lancer le service vsftpd dans le niveau 3 et 5**

**Suppression du service vsftpd.Le fichier /etc/init.d/vsftpd n’est supprimé, seuls les liens situés dans les répertoires /etc/rc.d/rcx.d sont supprimés**

[root@machine1 /]# **cd /etc/rc.d/rc5.d**

[root@machine1 rc5.d]# **ls -ls \*vsftpd\***

**0 lrwxrwxrwx. 1 root root 16 2 juin 15:27 S60vsftpd -> ../init.d/vsftpd**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **chkconfig --del vsftpd**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **ls -ls \*vsftpd\***

**ls: impossible d'accéder à \*vsftpd\*: Aucun fichier ou dossier de ce type**

[root@machine1 rc5.d]#

**Ajout du service vsftpd. Avec l’option add, il y aura vérification du script dans /etc/init.d**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **chkconfig --add vsftpd**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **ls -ls \*vsftpd\***

**0 lrwxrwxrwx. 1 root root 16 3 juin 09:53 K50vsftpd -> ../init.d/vsftpd**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **chkconfig --list | grep vsftpd**

**vsftpd 0:arrêt 1:arrêt 2:arrêt 3:arrêt 4:arrêt 5:arrêt 6:arrêt**

[root@machine1 rc5.d]#

**Le processus a bien été rajouté dans la liste avec l’option ‘arrêt’ pour tous les niveaux.**

**Lancement automatique du processus vsftpd aux niveaux 3 et 5**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **chkconfig --level 35 vsftpd on**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **chkconfig --list | grep vsftpd**

**vsftpd 0:arrêt 1:arrêt 2:arrêt 3:marche 4:arrêt 5:marche 6:arrêt**

[root@machine1 rc5.d]#

[root@machine1 rc5.d]# **ls -ls \*vsftpd\***

**0 lrwxrwxrwx. 1 root root 16 3 juin 09:54 S60vsftpd -> ../init.d/vsftpd**

[root@machine1 rc5.d]#

**Pour lancer (ou ne pas lancer) un processus au démarrage, on peut utiliser la commande ntsysv**

**La commande service.**

**On peut lancer ou arrêter un script qui est présent dans /etc/init.d**

[root@machine1 /]#

[root@machine1 /]# **service vsftpd stop**

**Arrêt de vsftpd : [ OK ]**

[root@machine1 /]#

[root@machine1 /]# **service vsftpd start**

**Démarrage de vsftpd pour vsftpd : [ OK ]**

[root@machine1 /]#

[root@machine1 /]# **/etc/init.d/vsftpd stop**

**Arrêt de vsftpd : [ OK ]**

[root@machine1 /]#

[root@machine1 /]# **/etc/init.d/vsftpd start**

**Démarrage de vsftpd pour vsftpd : [ OK ]**

[root@machine1 /]#

**On peut aussi utiliser le fichier /etc/init.d/rc.local pour lancer une commande qui sera exécutée au moment de l’initialisation du système.**

[root@machine1 rc.d]# **more rc.local**

**#!/bin/sh**

**#**

**# This script will be executed \*after\* all the other init scripts.**

**# You can put your own initialization stuff in here if you don't**

**# want to do the full Sys V style init stuff.**

**touch /var/lock/subsys/local**

[root@machine1 rc.d]#

**Ce fichier est souvent utilisé pour mettre à l’heure le serveur**

**/usr/sbin/ntpdate –s –b xxx.xxx.xxx.xxx**