



جامعة مولاي إسماعيل  
UNIVERSITÉ MOULAY ISMAÏL

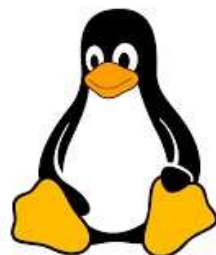


المدرسة العليا للأساتذة  
ÉCOLE NORMALE SUPÉRIEURE

# LES SERVICES LINUX "DAEMONS"

Réaliser par:

- Hrich khadija
- Chaouchi Meryeme
- EL Agri Manal



Encadré par:

D . MOUKHAFI MEHDI

# But & Application et expérimentation :

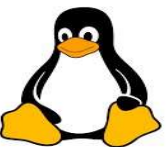
---

## ✓ But :

- Présentation des services (DAEMONS):
- Apprentissage de base pour bien comprendre les démons.

## ✓ Application et expérimentation :

- La création d'un démon.
- La lecture des journaux services .
- Comprendre Les commandes concernant ( services. , les unités de systemd , état de système ).



## 1- Services/Démons

-qu'est ce que les services et démon sur Linux .

-Les types de services /Démon.

## 2- Système d'initialisation .

## 3- Systemd / Systemctl.

## 4- Histoire.

## 5- Les composants de systemd .

## 6-les unités (Unit ) de systemd.

- Définition.

-Les types des unités.

- Les fichiers des unités.

-La relation entre unités et Fichier d'unités .

## 7- Architecture de démon.

# Plan :

## 8- Création d'un démon .

-Création d'un démon avec langage c.

- Installation d'un démon de système.

## 9- Gestion de service.

## 10-Présentation générale d'états de système.

## 11-Gestion d'unité.

## 12- Les journaux.

## 13- La sécurité des démons.

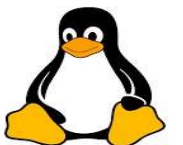


## Qu'est ce qu 'un service et demon sur l'Unix ?

Sur les systèmes Linux, les termes "services" et "daemons" (ou "démons") sont couramment utilisés pour désigner des programmes ou des processus en arrière-plan qui fournissent des fonctionnalités spécifiques au système. Bien que les deux termes soient souvent utilisés de manière interchangeable, ils ont des nuances légèrement différentes.

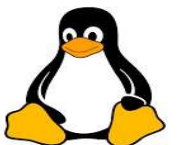
### Services :

Un service est un programme ou un processus qui s'exécute en arrière-plan et fournit des fonctionnalités spécifiques au système. Les services sont souvent conçus pour être démarrés au démarrage du système et fonctionnent en continu pour répondre à des demandes ou effectuer des tâches spécifiques. Les services peuvent inclure des démons, des serveurs web, des serveurs de bases de données, des serveurs de fichiers, etc. Ils sont souvent gérés par le système d'initialisation du système d'exploitation.



## Démons (Daemons) :

Un démon (daemon) est un type spécifique de programme en arrière-plan qui fonctionne de manière continue, sans interaction directe avec l'utilisateur. Les démons sont généralement des services système qui exécutent des tâches essentielles pour le fonctionnement du système. Les démons peuvent également être spécifiquement associés à des services, tels que le démon Apache (http) pour le serveur web Apache, le démon MySQL (MySQL) pour le système de gestion de base de données MySQL, etc.



## Les types de services en Linux :

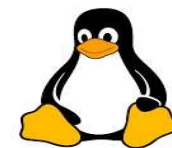
Sur les systèmes Linux, il existe différents types de services, chacun fournissant des fonctionnalités spécifiques au système. Ces services peuvent être classés en plusieurs catégories en fonction de leurs rôles et de leurs utilisations.

**Voici quelques types de services courants sur Linux :**

### Les services système essentiel:

**Serveurs SSH :**(Secure Shell) est un protocole de réseau utilisé pour sécuriser les communications sur un réseau

**Serveurs NTP :**Les serveurs NTP (Network Time Protocol) synchronisent l'horloge des systèmes sur un réseau .



# Services/Démons

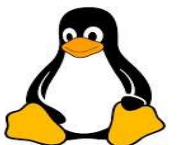
---

## Les services utilisateurs:

**Serveurs de Messagerie :** Des services comme (SMTP), fournissent des fonctionnalités de messagerie électronique.

**Serveurs VPN :** Le service VPN (Virtual Private Network) permettent la création de connexions sécurisées sur un réseau public.

**Serveurs d'Impression :** Des services tels que CUPS (Common Unix Printing System) gèrent l'impression sur le système.

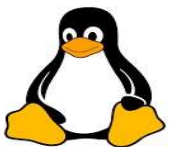


## Les services réseaux:

**Serveurs DHCP :** Les serveurs DHCP (Dynamic Host Configuration Protocol) fournissent des configurations réseau automatiques aux clients.

**Serveurs Web :** Les services web tels qu'Apache HTTP ou Nginx sont utilisés pour héberger des sites web et servir des pages web.

**Serveurs DNS :** Les serveurs DNS (Domain Name System) convertissent les noms de domaine en adresses IP.





## Les types des démons en Linux :

### Démons Système (System Daemons) :

#### Définition :

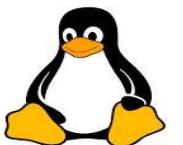
Les démons système sont des services qui s'exécutent en arrière-plan au niveau du système et ne sont associés à aucun utilisateur spécifique. Ils sont lancés au démarrage du système et gèrent des tâches essentielles du système d'exploitation.

#### Exemples :

**systemd** : Le démon système principal qui gère le processus d'initialisation et coordonne de nombreux aspects du système.

**udev** : Gère les périphériques du noyau et gère l'ajout ou la suppression dynamique des périphériques.

**NetworkManager** : Gère la connectivité réseau.



## Démons Utilisateurs (User Daemons) :

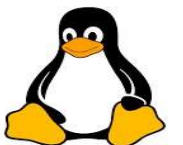
### Définition :

Les démons utilisateurs sont des services spécifiques à un utilisateur et s'exécutent en arrière-plan pour répondre à des besoins particuliers de cet utilisateur. Ils sont souvent lancés au moment de la connexion de l'utilisateur.

### Exemples :

**gnome-keyring-daemon** : Gère les clés et les mots de passe pour l'environnement de bureau .

**pulseaudio** : Gère le son au niveau de l'utilisateur.



# Système d'initialisation

---

## Système d'initialisation :

Le système d'initialisation est le premier processus qui démarre lorsqu'un système d'exploitation démarre. Son rôle principal est de démarrer et d'arrêter les processus et services du système. Il configure l'environnement initial du système, lance les processus nécessaires au démarrage et prend en charge la séquence de démarrage.

## Exemples de systèmes d'initialisation :

SysV init, Upstart, systemd.

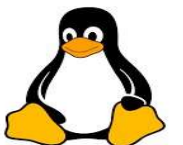


## Qu'est ce que systemd dans Linux ?

Systemd est un système d'initialisation moderne et puissant utilisé par de nombreuses distributions Linux pour gérer les processus, les services et les tâches au démarrage du système et tout au long de son fonctionnement. Il est conçu pour remplacer les systèmes d'initialisation traditionnels tels que SysVinit et apporte de nombreuses fonctionnalités avancées.

```
[manal@fedora ~]$ systemctl --version
systemd 253 (253.2-1.fc38)
+PAM +AUDIT +SELINUX +APPARMOR +IMA +SMACK +SECCOMP +GCRYPT +GNUTLS +OPENSSL +ACL +BLKID +CURL +ELFUTILS +FIDO2 +IDN2 -IDN -IPTC +KMOD +LIBCRYPTSETUP +LIBFDISK +PCRE2 +
PWQUALITY +P11KIT +QRENCODE +TPM2 +BZIP2 +LZ4 +XZ +ZLIB +ZSTD +BPF_FRAMEWORK +XKBCOMMON +UTMP +SYSVINIT default-hierarchy=unified
[manal@fedora ~]$ ps -p 1 -o comm=
systemd
[manal@fedora ~]$
```

Activate Windows  
Go to Settings to activate Windows.



# Systemd/Systemctl

## Définition:

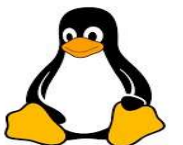
‘systemctl’ est une commande utilisée pour contrôler le système d'initialisation systemd, qui est largement utilisé sur de nombreuses distributions Linux.

‘systemctl’ permet aux utilisateurs de gérer les services, les unités et d'autres aspects du système qui sont sous le contrôle de systemd.

**Voici quelques utilisations courantes de systemctl :**

### Gestion des Services :

- Démarrer un service
- Arrêter un service
- Redémarrer un service
- Vérifier le statut d'un service
- Activer un service au démarrage
- Désactiver un service au démarrage

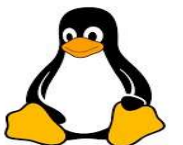


## Gestion des Unités :

- Afficher toutes les unités en cours d'exécution
- Afficher toutes les unités activées au démarrage
- Vérifier le statut d'une unité particulière.

## Gestion des Cibles (Target) :

- changer la cible du système ( par exemple, passer de graphical.target à multi-user.target ).



# Histoire

## Les débuts de Linux :

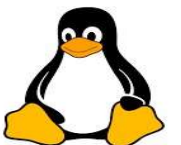
**1991 :** Linus Torvalds annonce la création du noyau Linux. Le terme "démon" est utilisé pour désigner des processus en arrière-plan fournissant des services au système, sans connotation maléfique.

## Les premiers jours et Tux :

**1996 :** Le pingouin Tux devient la mascotte non officielle de Linux. Bien que l'histoire du démon ait été mentionnée par Linus Torvalds, le terme "daemon" reste associé aux processus en arrière-plan, sans connotation maléfique.

## Évolution des systèmes d'initialisation :

**Années 1990-2000 :** Les systèmes Linux utilisent généralement le système d'initialisation SysV init pour gérer les processus au démarrage du système.



# Histoire

---

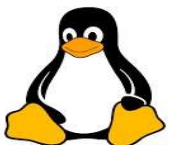
## Introduction de systemd et gestion des démons :

**2010 :** systemd est introduit comme un système d'initialisation moderne. Il prend en charge la gestion des démons système, fournissant des fonctionnalités avancées pour le contrôle et la supervision des services.

## Développements récents :

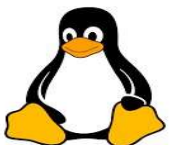
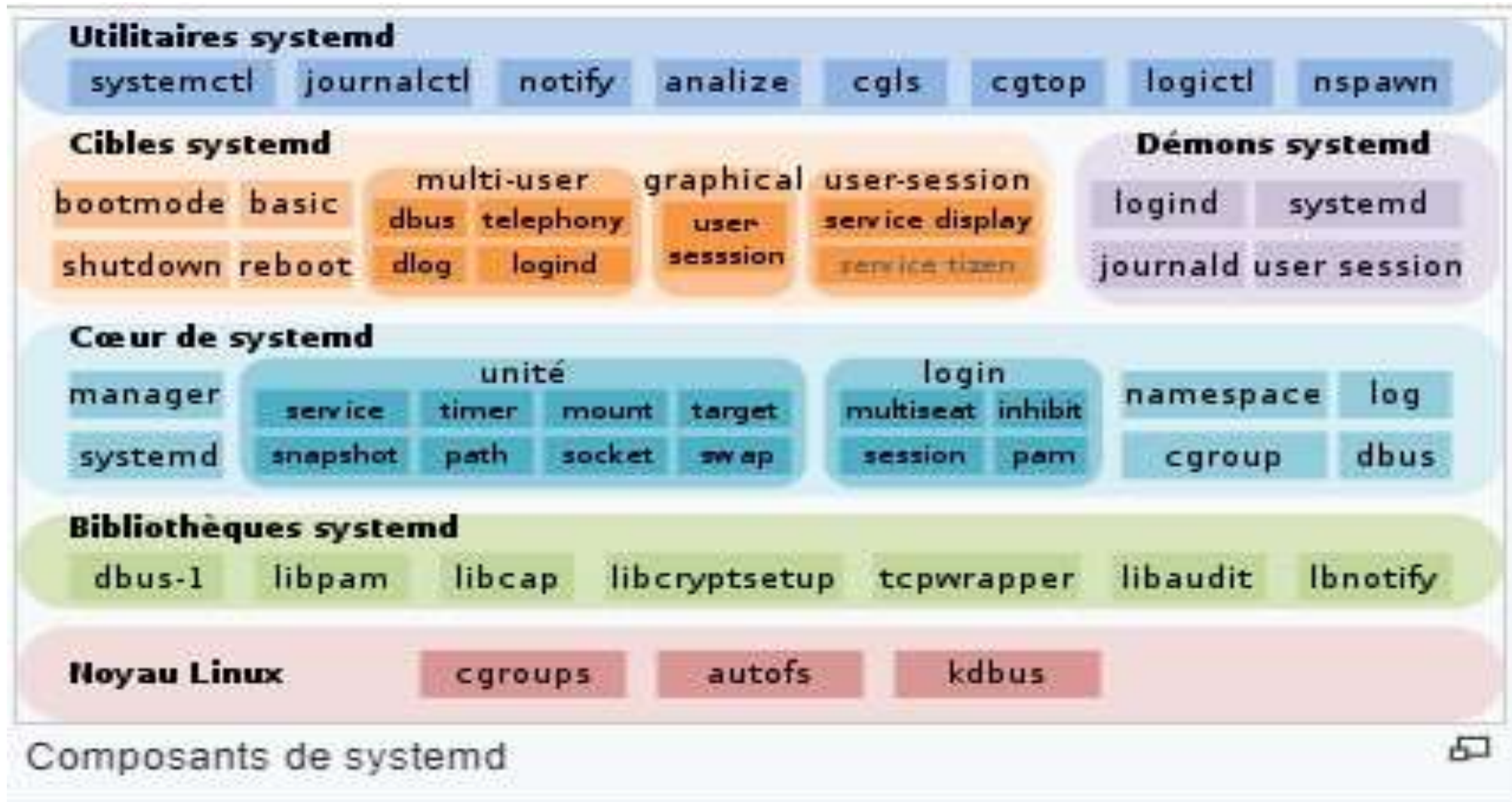
**2015:** le projet systemd a créé des discussions au sein de la communauté Linux en raison de ses méthodes modernes et parfois contestées pour gérer les services système.

**2016:** les débats concernant systemd et d'autres aspects du système d'exploitation Linux ont persisté, mettant en lumière la diversité des opinions au sein de la communauté open source.





# Les composants de systemd



# les unités (Unit) systemd

## Définition:

Dans le contexte de systemd, une "unit" (unité) fait référence à une unité de configuration qui représente un objet ou un service géré par le système d'initialisation systemd. Les unités sont utilisées pour décrire et configurer divers aspects du système et des services.

## Les types des unités:

**Service Units (.service)** : Les unités de service représentent des services ou des démons spécifiques qui s'exécutent en arrière-plan. Par exemple, le service Apache HTTP Server a une unité de service appelée apache2.service.

**Socket Units (.socket)** : Ces unités représentent des points de connexion réseau (sockets) utilisés pour la communication interprocessus. Elles permettent de démarrer un service uniquement lorsque la connexion est établie.



# les unités (Unit) systemd

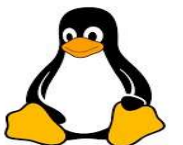
**Target Units (.target) :** Les unités de cible regroupent plusieurs autres unités et définissent un état du système. Par exemple, la cible multi-user. Target regroupe les unités nécessaires pour démarrer en mode multi-utilisateur.

**Device Units (.device) :** Ces unités représentent des périphériques matériels. systemd peut gérer l'activation et la désactivation dynamique de ces périphériques.

**Mount Units (.mount) :** Les unités de montage représentent des points de montage de systèmes de fichiers. Elles spécifient comment monter des systèmes de fichiers lors du démarrage.

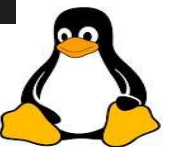
**Automount Units (.automount) :** Ces unités décrivent les points de montage de systèmes de fichiers automatiques, qui ne sont montés que lorsqu'ils sont accédés.

**Timer Units (.timer) :** Les unités de minuterie permettent de déclencher l'exécution périodique d'unités de service.



# les unités (Unit) systemd

```
[manal@fedora ~]$ systemctl -t help
Available unit types:
service
mount
swap
socket
target
device
automount
timer
path
slice
scope
[manal@fedora ~]$
```



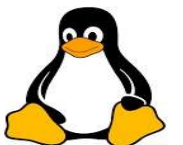
# les unités (Unit) systemd

## Fichiers d'Units :

Les fichiers d'unités sont des fichiers de configuration qui spécifient comment systemd doit gérer une unité spécifique. Ces fichiers sont généralement situés dans le répertoire `/etc/systemd/system/` pour les configurations locales et `/usr/lib/systemd/system/` pour les configurations système.

## Relation entre Units et Fichiers d'Units :

Chaque unité systemd est associée à un fichier d'unité correspondant. Le fichier d'unité définit la configuration spécifique pour cette unité. Lorsque vous interagissez avec systemd pour gérer une unité (par exemple, démarrer ou arrêter un service), systemd se réfère au fichier d'unité pour comprendre comment effectuer ces actions.



# Architecture du démon.

**L'architecture d'un démon** (ou daemon en anglais) sur un système Unix-like est généralement conçue pour qu'il puisse s'exécuter en arrière-plan de manière continue, sans nécessiter d'interaction directe avec l'utilisateur.

**Voici une explication détaillée de l'architecture typique d'un démon :**

## **1\_Processus Démon :**

- Un démon est un processus qui s'exécute en arrière-plan de manière continue. Il est généralement lancé au démarrage du système et reste actif jusqu'à ce que le système soit éteint.

## **2\_Création d'un Processus Enfant (Forking) :**

- Pour devenir un démon, le processus initial se duplique en créant un processus enfant à l'aide de la fonction système `fork()`. Le processus parent se termine ensuite, laissant le processus enfant indépendant.



# Architecture du démon.

## 3\_Changement de Session (setsid) :

- Le processus enfant change de session en appelant la fonction ``setsid()``. Cela détache le processus du terminal d'origine, assurant ainsi son indépendance.

## 4\_Ignorer le Signal SIGHUP :

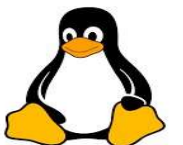
- Pour éviter que le démon ne se ferme lorsque la session de l'utilisateur se termine, le démon ignore le signal SIGHUP en appelant ``signal(SIGHUP, SIG_IGN)``.

## 5\_Changement de Répertoire de Travail :

- Le démon change son répertoire de travail vers un répertoire sûr, généralement le répertoire racine (``chdir("/")``). Cela garantit que le démon n'est pas lié à un répertoire spécifique du système de fichiers.

## 6\_Fermeture des Descripteurs de Fichiers Hérités :

- Le démon ferme tous les descripteurs de fichiers hérités du processus parent, tels que stdin, stdout et stderr (``close(STDIN_FILENO)``, ``close(STDOUT_FILENO)``, ``close(STDERR_FILENO)``).





# Architecture du démon.

## 7\_Exécution du Code du Démon :

- Après ces étapes, le démon exécute le code spécifique du démon. Cela peut inclure la gestion des tâches planifiées, la surveillance de l'état du système, la fourniture de services, etc.

## 8\_Boucle Infinie (Optionnelle) :

- Souvent, un démon exécute une boucle infinie pour rester actif et répondre aux événements ou aux requêtes. Par exemple, il pourrait attendre des connexions réseau, surveiller des répertoires, ou effectuer d'autres tâches périodiques.

## 9\_Gestion des Signaux :

- Le démon peut également inclure une gestion des signaux pour réagir à certains événements.

## 10\_Fermeture Propre :

- Lorsque le démon est arrêté, il doit effectuer une fermeture propre, libérant les ressources, fermant les connexions réseau et effectuant d'autres tâches nécessaires avant de se terminer.



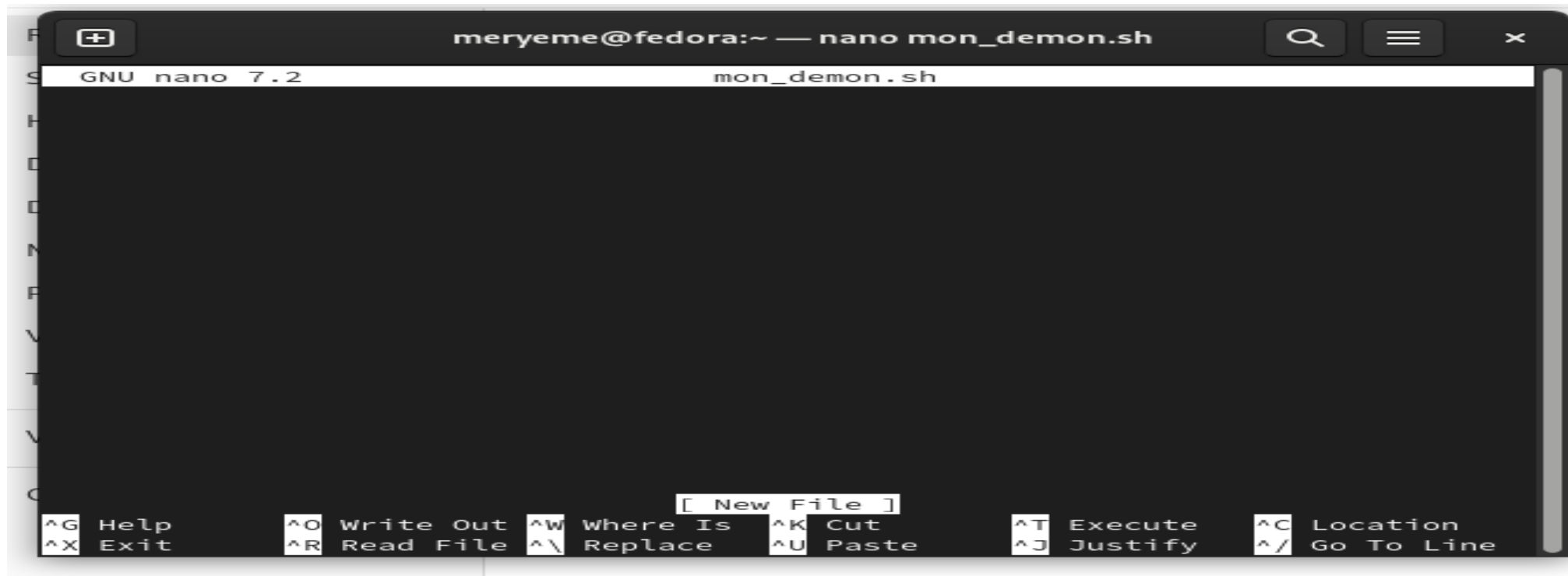
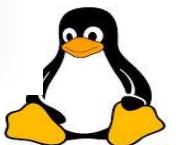


# Création d'un démon

## Création d'un démon avec langage c:

**1- Ecriture du code:** Ouvrez un éditeur de texte (comme nano , vim, gedit,etc) et copier le code c dans un nouveau fichier . Sauvegarder le fichier avec une extension '.c'

`nano mon_demon.c`

A screenshot of the nano text editor interface. The title bar at the top shows the user 'meryeme@fedora:~' and the file name 'nano mon\_demon.sh'. Below the title bar, the text 'GNU nano 7.2' and 'mon\_demon.sh' are visible. The main editing area is empty. At the bottom, a status bar displays various keyboard shortcuts: '^G Help', '^O Write Out', '^W Where Is', '^K Cut', '^T Execute', '^C Location', '^X Exit', '^R Read File', '^\_ Replace', '^U Paste', '^J Justify', and '^/ Go To Line'. A '[ New File ]' prompt is also visible in the center of the status bar.

# Création d'un démon

## 2\_ le code en nano:

```
manal@fedora:~ — nano mon_demon.c
GNU nano 7.2 mon_demon.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>

void signal_handler(int signo){
    if (signo == SIGHUP) {
        //Gerer le signal SIGHUP (par exemple, recharger la configuration)
        printf("Recharger la configuration...\n");
    }
}

int main(){
    //Creer un processus enfant
    pid_t pid = fork();
    //verifier s'il y a une erreur lors de la creation du processus
    if (pid < 0) {
        perror("Erreur lors de la creation du processus");
        exit(EXIT_FAILURE);
    }

    //Si le processus parent, terminer
    if (pid > 0) {
        exit(EXIT_SUCCESS);
    }
}
```

^G Aide    ^O Écrire    ^W Chercher    ^K Couper    ^T Exécuter    ^C Emplacement    M-U Annuler    M-A Marquer    M-J -> Crochet    M-Q Précédent  
^X Quitter    ^R Lire fich.    ^\ Remplacer    ^U Coller    ^J Justifier    ^/ Aller ligne    M-E Refaire    M-6 Copier    ^Q Retrouver    M-W Suivant



# Création d'un démon

## 2\_ le code en nano:

```
manal@fedora:~ — nano mon_demon.c
GNU nano 7.2 mon_demon.c

//Changer le masque de creation de fichier (umask)
umask(0);

//Creer une nouvelle session de groupe
if (setsid() < 0) {
perror("Erreur lors de la creation de la nouvelle session de groupe");
exit(EXIT_FAILURE);
}

//Ignorer le signal SIGHUP
signal(SIGHUP, SIG_IGN);

//Gerer le signal de SIGHUP
signal(SIGHUP, signal_handler);

//Changer le repertoire de travail vers le repertoire racine
if (chdir("/") < 0){
perror("Erreur lors du changement de repertoire de travail");
exit(EXIT_FAILURE);
}

//Fermer les descripteurs de fichiers herites
close(STDIN_FILENO);
close(STDOUT_FILENO);
close(STDERR_FILENO);

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement M-U Annuler   M-A Marquer   M-J -> Crochet M-Q Précédent
^X Quitter   ^R Lire fich.^_ Remplacer  ^U Coller    ^J Justifier ^/ Aller ligne M-E Refaire   M-6 Copier    ^Q Retrouver M-W Suivant
```



# Création d'un démon

## 2\_ le code en nano:

```
manal@fedora:~ — nano mon_demon.c
GNU nano 7.2 mon_demon.c
signal(SIGHUP, SIG_IGN);

//Gerer le signal de SIGHUP
signal(SIGHUP, signal_handler);

//Changer le repertoire de travail vers le repertoire racine
if (chdir("/") < 0){
perror("Erreur lors du changement de repertoire de travail");
exit(EXIT_FAILURE);
}

//Fermer les descripteurs de fichiers herites
close(STDIN_FILENO);
close(STDOUT_FILENO);
close(STDERR_FILENO);

//Executer le code du demon ici...

//Exemple : Boucle infinie pour maintenir le demon en vie
while (1) {
    //code du demon
    sleep(1);
}
return 0;
}
```

^G Aide    ^O Écrire    ^W Chercher    ^K Couper    ^T Exécuter    ^C Emplacement    M-U Annuler    M-A Marquer    M-] -> Crochet    M-O Précédent  
^X Quitter    ^R Lire fich.    ^\_ Remplacer    ^U Coller    ^J Justifier    ^/ Aller ligne    M-E Refaire    M-6 Copier    ^Q Retrouver    M-W Suivant



# Création d'un démon

## 2\_ Enregistrer le fichier :

dans nano en utilisant la commande **ctrl+o** , puis **entrer**.

## 3-Quittez nano:

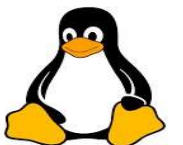
en utilisant la commande **ctrl+x** ,

## 4- Compilez le code c:

pour compiler le code en utilisant la commande :

```
gcc -o mon_demon mon_demon.c
```

➡ La commande indique au compilateur GCC de prendre le fichier source , de la compiler , et de générer un exécutable 'mon\_demon' . Après avoir exécuté cette commande avec succès , vous obtiendrez un fichier exécutable ('mon\_demon') qui vous pourrez ensuite exécuter sur votre système.



# Création d'un démon

## Attention!

le commande gcc est commande de compilation en langage c .

Pour vérifier si GCC est installé, ouvrez un terminal et exécutez la commande suivante :

**gcc --version**

```
meryeme@fedora:~  
[meryeme@fedora ~]$ gcc --version  
gcc (GCC) 13.2.1 20231011 (Red Hat 13.2.1-4)  
Copyright (C) 2023 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions.  There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
[meryeme@fedora ~]$
```



# Création d'un démon

Si ce n'est pas installé, vous pouvez l'installer en utilisant la commande suivante dans le terminal : **sudo dnf install gcc**

```
meryeme@fedora:~ — sudo dnf install gcc
```

Package	Architecture	Version	Repository	Size
Installing:				
gcc	x86_64	13.2.1-4.fc38	updates	34 M
Upgrading:				
cpp	x86_64	13.2.1-4.fc38	updates	11 M
libgcc	x86_64	13.2.1-4.fc38	updates	111 k
libgomp	x86_64	13.2.1-4.fc38	updates	320 k
Installing dependencies:				
binutils	x86_64	2.39-9.fc38	fedora	5.4 M
binutils-gold	x86_64	2.39-9.fc38	fedora	784 k
gc	x86_64	8.2.2-3.fc38	fedora	110 k
glibc-devel	x86_64	2.37-1.fc38	fedora	50 k
glibc-headers-x86	noarch	2.37-1.fc38	fedora	530 k
guile22	x86_64	2.2.7-7.fc38	fedora	6.5 M
kernel-headers	x86_64	6.5.4-200.fc38	updates	1.5 M
libxcrypt-devel	x86_64	4.4.33-7.fc38	fedora	30 k
make	x86_64	1:4.4.1-1.fc38	updates	588 k

Transaction Summary





# Création d'un démon

```
meryeme@fedora:~  
Verifying      : gc-8.2.2-3.fc38.x86_64 3/16  
Verifying      : glibc-devel-2.37-1.fc38.x86_64 4/16  
Verifying      : glibc-headers-x86-2.37-1.fc38.noarch 5/16  
Verifying      : guile22-2.2.7-7.fc38.x86_64 6/16  
Verifying      : libxcrypt-devel-4.4.33-7.fc38.x86_64 7/16  
Verifying      : gcc-13.2.1-4.fc38.x86_64 8/16  
Verifying      : kernel-headers-6.5.4-200.fc38.x86_64 9/16  
Verifying      : make-1:4.4.1-1.fc38.x86_64 10/16  
Verifying      : cpp-13.2.1-4.fc38.x86_64 11/16  
Verifying      : cpp-13.0.1-0.12.fc38.x86_64 12/16  
Verifying      : libgcc-13.2.1-4.fc38.x86_64 13/16  
Verifying      : libgcc-13.0.1-0.12.fc38.x86_64 14/16  
Verifying      : libgomp-13.2.1-4.fc38.x86_64 15/16  
Verifying      : libgomp-13.0.1-0.12.fc38.x86_64 16/16  
  
Upgraded:  
  cpp-13.2.1-4.fc38.x86_64      libgcc-13.2.1-4.fc38.x86_64      libgomp-13.2.1-4.fc38.x86_64  
Installed:  
  binutils-2.39-9.fc38.x86_64  binutils-gold-2.39-9.fc38.x86_64  gc-8.2.2-3.fc38.x86_64  gcc-13.2.1-4.fc38.x86_64  
  glibc-devel-2.37-1.fc38.x86_64  glibc-headers-x86-2.37-1.fc38.noarch  guile22-2.2.7-7.fc38.x86_64  kernel-headers-6.5.4-200.fc38.x86_64  
  libxcrypt-devel-4.4.33-7.fc38.x86_64  make-1:4.4.1-1.fc38.x86_64  
  
Complete!  
[meryeme@fedora ~]$
```





# Création d'un démon

## 5- Exécutez le programme :

avec la commande `./mon_demon`

Lorsque votre programme est conçu pour s'exécuter , vous pouvez également lancer le programme en arrière plan avec `'&'` `./mon_demon &`

```
[manal@fedora ~]$ nano mon_demon.c
[manal@fedora ~]$ ./mon_demon
[manal@fedora ~]$ ./mon demon &
[1] 3340
bash: ./mon: Aucun fichier ou dossier de ce type
[1]+  Termine 127          ./mon demon
[manal@fedora ~]$ ./mon_demon &
[1] 3345
[1]+  Fini                ./mon_demon
[manal@fedora ~]$ ls
A2      Cours      exo_linux  Modèles  mon_demon.c  projets  Téléchargements  testB.txt  tpl
Bureau  Documents  Images    mon_demon  Musique      Public    testA.txt        testC.txt  Vidéos
[manal@fedora ~]$
```



# Création d'un démon

## Installation d'un démon de système:

Pour installer le demon SHH (sshd) comme exemple en fedora , suivez les étapes :

### 1.Mettez à jour le référentiel de packages:

`sudo dnf update`

```
meryeme@fedora:~ — sudo dnf update
perl-locale                               noarch 1.10-497.fc38      updates 15 k
pipewire-jack-audio-connection-kit-libs   x86_64 0.3.85-1.fc38       updates 140 k
python3-urllib3+socks                     noarch 1.26.18-1.fc38       updates 9.6 k
qt5-qtgraphicaleffects                     x86_64 5.15.11-1.fc38       updates 120 k
qt5-qtquickcontrols2                       x86_64 5.15.11-1.fc38       updates 1.7 M
qt5-qtsvg                                  x86_64 5.15.11-1.fc38       updates 186 k
realtek-firmware                           noarch 20231030-1.fc38      updates 2.4 M
swtpm-selinux                             noarch 0.8.1-1.fc38         updates 20 k
xcb-util-cursor                            x86_64 0.1.4-2.fc38         fedora 19 k
Installing weak dependencies:
google-noto-sans-mono-cjk-vf-fonts         noarch 1:2.004-5.fc38       updates 14 M
gststreamer1-plugins-good-qt6              x86_64 1.22.5-3.fc38        updates 43 k
qadwaitadecorations-qt5                   x86_64 0.1.3-2.fc38         updates 58 k
qadwaitadecorations-qt6                   x86_64 0.1.3-2.fc38         updates 64 k
sqlite                                      x86_64 3.40.1-2.fc38        fedora 813 k

Transaction Summary
-----
Install   37 Packages
Upgrade  815 Packages

Total download size: 1.3 G
Is this ok [y/N]: y
```

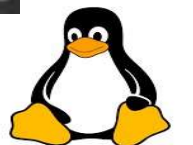


# Création d'un démon

## 2. Installez le package openssh-server:

**sudo dnf install openssh-server**

```
KeyboardInterrupt: Terminated.  
[meryeme@fedora ~]$ sudo dnf install openssh-server  
[sudo] password for meryeme:  
Last metadata expiration check: 3:16:20 ago on Mon 20 Nov 2023 06:51:32 AM EST.  
Package openssh-server-9.0p1-14.fc38.1.x86_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
[meryeme@fedora ~]$
```



# Gestion de service

## 1\_Démarrage et arrêt des services:

Si vous souhaitez démarrer un service systemd en exécutant les instructions qui se trouvent dans le fichier de l'unité du service, utilisez la commande **start**.

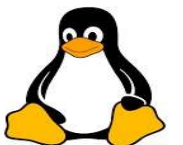
**systemctl start <nom de demon>**

```
[manal@fedora ~]$ systemctl start sshd.service  
[manal@fedora ~]$ systemctl start sshd
```

Pour arrêter un service en cours d'exécution, vous pouvez plutôt utiliser la commande **stop**

**systemctl stop <nom de demon>**

```
[manal@fedora ~]$ sudo systemctl stop sshd  
[sudo] Mot de passe de manal :  
[manal@fedora ~]$
```



# Gestion de service

## 2-Redémarrage et rechargement:

Pour redémarrer un service en cours d'exécution, vous pouvez utiliser la commande **restart**

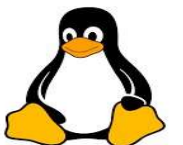
**systemctl restart <nom de demon>**

```
[manal@fedora ~]$ sudo systemctl restart sshd  
[manal@fedora ~]$
```

Si l'application en question est en capacité de recharger ses fichiers de configuration (sans redémarrage), vous pouvez lancer la commande **reload** pour initier ce processus

**systemctl reload <nom de demon>**

```
[manal@fedora ~]$ sudo systemctl reload sshd  
[manal@fedora ~]$
```



# Gestion de service

Si vous ne savez pas si le service intègre la fonctionnalité qui lui permet de recharger sa configuration, vous pouvez lancer la commande **reload-or-restart**. Si disponible, vous rechargerez la configuration en place. Sinon, le service redémarrera pour récupérer la nouvelle configuration

**systemctl reload-or-restart <nom de demon>**

```
[manal@fedora ~]$ sudo systemctl reload-or-restart sshd  
[manal@fedora ~]$
```

## 3-Activation et désactivation des services:

Les commandes ci-dessus vous seront utiles pour démarrer ou arrêter des services pendant la session en cours. Vous devez les activer pour demander à systemd de lancer automatiquement les services au démarrage. Pour lancer un service au démarrage, utilisez la commande **enable**

**systemctl enable <nom de demon>**



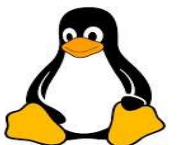
# Gestion de service

```
[manal@fedora ~]$ sudo systemctl enable sshd
```

Pour désactiver le démarrage automatique d'un service, vous pouvez saisir la commande **disable** `systemctl disable <nom de demon>`

```
[manal@fedora ~]$ sudo systemctl disable sshd.service  
[manal@fedora ~]$
```

N'oubliez pas que l'activation du service ne le déclenche pas pendant la session en cours. Si vous souhaitez démarrer le service et, dans le même temps, l'activer au démarrage, vous devez lancer à la fois la commande `start` et la commande `enable`





# Gestion de service

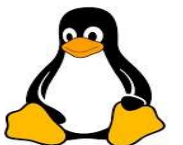
## Vérification de l'état des services:

Pour vérifier l'état d'un service sur votre système, vous pouvez utiliser la commande **status**

**systemctl status <nom de demon>**

```
[manal@fedora ~]$ systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/service.d
            └─10-timeout-abort.conf
   Active: active (running) since Sun 2023-11-19 12:20:14 EST; 10min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 5184 (sshd)
    Tasks: 1 (limit: 1408)
   Memory: 1.3M
      CPU: 166ms
   CGroup: /system.slice/sshd.service
           └─5184 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

nov. 19 12:22:45 fedora systemd[1]: Reloading sshd.service - OpenSSH server daemon...
nov. 19 12:22:45 fedora sshd[5184]: Received SIGHUP; restarting.
nov. 19 12:22:45 fedora systemd[1]: Reloaded sshd.service - OpenSSH server daemon.
nov. 19 12:22:45 fedora sshd[5184]: Server listening on 0.0.0.0 port 22.
nov. 19 12:22:45 fedora sshd[5184]: Server listening on :: port 22.
nov. 19 12:24:39 fedora systemd[1]: Reloading sshd.service - OpenSSH server daemon...
nov. 19 12:24:39 fedora sshd[5184]: Received SIGHUP; restarting.
nov. 19 12:24:39 fedora systemd[1]: Reloaded sshd.service - OpenSSH server daemon.
nov. 19 12:24:39 fedora sshd[5184]: Server listening on 0.0.0.0 port 22.
nov. 19 12:24:39 fedora sshd[5184]: Server listening on :: port 22.
[manal@fedora ~]$
```





# Gestion de service

Cela vous donne un bon aperçu de l'état actuel de l'application, vous signalant tout problème et toute action à mettre en œuvre.

Certaines méthodes vous permettent également de contrôler des états spécifiques. Par exemple, si vous souhaitez vérifier si une unité est actuellement active (en cours d'exécution), vous pouvez utiliser la commande

**is-active**                    **systemctl is-active <nom de demon>**

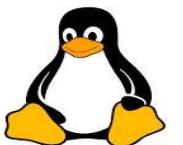
```
[manal@fedora ~]$ systemctl is-active sshd.service
active
[manal@fedora ~]$
```

Cela renverra l'état actuel de l'unité, qui est généralement active ou inactive.

Pour voir si l'unité est activée, vous pouvez utiliser la commande **is-enabled**

**systemctl is-enabled <nom de demon>**

```
[manal@fedora ~]$ systemctl is-enabled sshd.service
disabled
[manal@fedora ~]$
```



# Présentation générale d'états de système

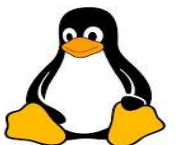
Jusqu'à présent, les commandes nous ont permis de gérer des services uniques, mais pas vraiment de consulter l'état actuel du système. Il existe un certain nombre de commandes `systemctl` qui vous donnent ces informations.

Liste des unités en cours d'utilisation .

Pour avoir une liste de toutes les unités actives que `systemd` reconnaît, nous pouvons utiliser la commande **list-units** : `systemctl list-units`

```
[manal@fedora ~]$ systemctl list-units
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
proc-sys-fs-binfmt_misc.automount	loaded	active	running	Arbitrary Executable File Formats File System Autom...
sys-devices-pci0000:00-0000:00:01.1-ata3-host2-target2:0:0-2:0:0:0-block-sr0.device	loaded	active	plugged	VBOX_CD-ROM
sys-devices-pci0000:00-0000:00:02.0-drm-card0.device	loaded	active	plugged	/sys/devices/pci0000:00/0000:00:02.0/drm/card0
sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device	loaded	active	plugged	82540EM Gigabit Ethernet Controller (PRO/1000 MT De...
sys-devices-pci0000:00-0000:00:05.0-sound-card0-controlC0.device	loaded	active	plugged	/sys/devices/pci0000:00/0000:00:05.0/sound/card0/co...
sys-devices-pci0000:00-0000:00:0d.0-ata1-host0-target0:0:0-0:0:0:0-block-sda-sda1.device	loaded	active	plugged	VBOX_HARDDISK 1
sys-devices-pci0000:00-0000:00:0d.0-ata1-host0-target0:0:0-0:0:0:0-block-sda-sda2.device	loaded	active	plugged	VBOX_HARDDISK 2
sys-devices-pci0000:00-0000:00:0d.0-ata1-host0-target0:0:0-0:0:0:0-block-sda-sda3.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
sys-devices-pci0000:00-0000:00:0d.0-ata1-host0-target0:0:0-0:0:0:0-block-sda.device	loaded	active	plugged	VBOX_HARDDISK
sys-devices-platform-serial8250-tty-ttyS0.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS0
sys-devices-platform-serial8250-tty-ttyS1.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS1
sys-devices-platform-serial8250-tty-ttyS10.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS10
sys-devices-platform-serial8250-tty-ttyS11.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS11
sys-devices-platform-serial8250-tty-ttyS12.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS12
sys-devices-platform-serial8250-tty-ttyS13.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS13
sys-devices-platform-serial8250-tty-ttyS14.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS14
sys-devices-platform-serial8250-tty-ttyS15.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS15
sys-devices-platform-serial8250-tty-ttyS16.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS16
sys-devices-platform-serial8250-tty-ttyS17.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS17
sys-devices-platform-serial8250-tty-ttyS18.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS18
sys-devices-platform-serial8250-tty-ttyS19.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS19
sys-devices-platform-serial8250-tty-ttyS2.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS2
sys-devices-platform-serial8250-tty-ttyS20.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS20
sys-devices-platform-serial8250-tty-ttyS21.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS21
sys-devices-platform-serial8250-tty-ttyS22.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS22
sys-devices-platform-serial8250-tty-ttyS23.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS23
sys-devices-platform-serial8250-tty-ttyS24.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS24
sys-devices-platform-serial8250-tty-ttyS25.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS25
sys-devices-platform-serial8250-tty-ttyS26.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS26
sys-devices-platform-serial8250-tty-ttyS27.device	loaded	active	plugged	/sys/devices/platform/serial8250/tty/ttyS27



# Présentation générale d'états de système

**La sortie affiche les colonnes suivantes :**

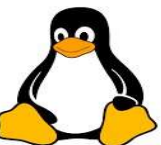
**UNIT** : le nom de l'unité systemd

**LOAD** : si la configuration de l'unité a été analysée par systemd. La configuration des unités chargées est gardée en mémoire.

**ACTIVE** : un état résumé indiquant si l'unité est active. Il s'agit généralement d'une méthode assez simple d'établir si l'unité a bien démarré ou pas.

**SUB** : il s'agit d'un état de niveau inférieur qui donne des informations plus détaillées sur l'unité. Il varie souvent en fonction du type, de l'état et du mode de fonctionnement réel de l'unité.

**DESCRIPTION** : une courte description textuelle de ce que l'unité est/fait.

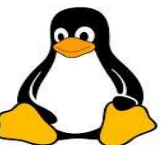


# Présentation générale d'états de système

Nous pouvons instruire `systemctl` de générer des informations différentes en ajoutant des balises supplémentaires. Par exemple, pour consulter toutes les unités que `systemd` a chargées (ou tente de charger), qu'elles soient actuellement actives ou pas, vous pouvez utiliser la balise `--all` comme suit : `systemctl list-units --all`

```
[manal@fedora ~]$ systemctl list-units --all
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
boot.automount	not-found	inactive	dead	boot.automount
proc-sys-fs-binfmt_misc.automount	loaded	active	running	Arbitrary Executable File Formats File System
dev-cdrom.device	loaded	active	plugged	VBOX_CD-ROM
dev-disk-by\x2ddiskseq-1.device	loaded	active	plugged	VBOX_HARDDISK
dev-disk-by\x2ddiskseq-1\x2dpart1.device	loaded	active	plugged	VBOX_HARDDISK 1
dev-disk-by\x2ddiskseq-1\x2dpart2.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2ddiskseq-1\x2dpart3.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2ddiskseq-3.device	loaded	active	plugged	VBOX_CD-ROM
dev-disk-by\x2did-ata\x2dVBOX_CD\x2dROM_VB2\x2d01700376.device	loaded	active	plugged	VBOX_CD-ROM
dev-disk-by\x2did-ata\x2dVBOX_HARDDISK_VBe079d01c\x2d3edf9773.device	loaded	active	plugged	VBOX_HARDDISK
dev-disk-by\x2did-ata\x2dVBOX_HARDDISK_VBe079d01c\x2d3edf9773\x2dpart1.device	loaded	active	plugged	VBOX_HARDDISK 1
dev-disk-by\x2did-ata\x2dVBOX_HARDDISK_VBe079d01c\x2d3edf9773\x2dpart2.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2did-ata\x2dVBOX_HARDDISK_VBe079d01c\x2d3edf9773\x2dpart3.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2dlabel-fedora_localhost\x2dlive.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2dpartuuid-447f654e\x2d02e3\x2d4a09\x2d95fa\x2d92a13d672ede.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2dpartuuid-674df1d4\x2d5ff1\x2d4fb8\x2d9c9d\x2d74995fe20414.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2dpartuuid-ae5c87be\x2dc62d\x2d4ab7\x2da57c\x2d71ea52c54497.device	loaded	active	plugged	VBOX_HARDDISK 1
dev-disk-by\x2dpath-pci\x2d0000:00:01.1\x2data\x2d2.0.device	loaded	active	plugged	VBOX_CD-ROM
dev-disk-by\x2dpath-pci\x2d0000:00:01.1\x2data\x2d2.device	loaded	active	plugged	VBOX_CD-ROM
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1.0.device	loaded	active	plugged	VBOX_HARDDISK
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1.0\x2dpart1.device	loaded	active	plugged	VBOX_HARDDISK 1
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1.0\x2dpart2.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1.0\x2dpart3.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1.device	loaded	active	plugged	VBOX_HARDDISK
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1\x2dpart1.device	loaded	active	plugged	VBOX_HARDDISK 1
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1\x2dpart2.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2dpath-pci\x2d0000:00:0d.0\x2data\x2d1\x2dpart3.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live
dev-disk-by\x2duuid-f7bfdce1\x2defea\x2d4120\x2dbbab\x2d7a7b3b650ad5.device	loaded	active	plugged	VBOX_HARDDISK 2
dev-disk-by\x2duuid-fdd13e5c\x2d0405\x2d45bc\x2d96c9\x2dcfdffb6cc66d.device	loaded	active	plugged	VBOX_HARDDISK fedora_localhost-live





# Présentation générale d'états de système

Vous pouvez filtrer ces résultats en utilisant d'autres balises. Par exemple, nous pouvons utiliser la balise `--state=` pour indiquer les états LOAD, ACTIVE ou SUB que nous souhaitons consulter. Nous allons devoir garder la balise `--all` pour que `systemctl` permette l'affichage des unités inactives : **`systemctl list-units --all --state=`**

```
[manal@fedora ~]$ systemctl list-units --all --state=inactive
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
boot.automount	not-found	inactive	dead	boot.automount
proc-fs-nfsd.mount	loaded	inactive	dead	NFSD configuration filesystem
sysroot.mount	not-found	inactive	dead	sysroot.mount
var-lib-machines.mount	loaded	inactive	dead	Virtual Machine and Container Storage (Compatibility)
var.mount	not-found	inactive	dead	var.mount
systemd-ask-password-console.path	loaded	inactive	dead	Dispatch Password Requests to Console Directory Watch
abrt-vmcore.service	loaded	inactive	dead	ABRT kernel panic detection
alsa-restore.service	loaded	inactive	dead	Save/Restore Sound Card State
apparmor.service	not-found	inactive	dead	apparmor.service
auth-rpcgss-module.service	loaded	inactive	dead	Kernel Module supporting RPCSEC_GSS
auto-cpufreq.service	not-found	inactive	dead	auto-cpufreq.service
dm-event.service	loaded	inactive	dead	Device-mapper event daemon
dnf-makecache.service	loaded	inactive	dead	dnf makecache
dracut-cmdline.service	loaded	inactive	dead	dracut cmdline hook
dracut-initqueue.service	loaded	inactive	dead	dracut initqueue hook
dracut-mount.service	loaded	inactive	dead	dracut mount hook
dracut-pre-mount.service	loaded	inactive	dead	dracut pre-mount hook
dracut-pre-pivot.service	loaded	inactive	dead	dracut pre-pivot and cleanup hook
dracut-pre-trigger.service	loaded	inactive	dead	dracut pre-trigger hook
dracut-pre-udev.service	loaded	inactive	dead	dracut pre-udev hook
dracut-shutdown-onfailure.service	loaded	inactive	dead	Service executing upon dracut-shutdown failure to perform
ebtables.service	not-found	inactive	dead	ebtables.service
emergency.service	loaded	inactive	dead	Emergency Shell
flatpak-add-fedora-repos.service	loaded	inactive	dead	Add Fedora flatpak repositories
flatpak-system-helper.service	loaded	inactive	dead	flatpak system helper
fstrim.service	loaded	inactive	dead	Discard unused blocks on filesystems from /etc/fstab
getty@tty1.service	loaded	inactive	dead	Getty on tty1

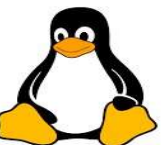


# Présentation générale d'états de système

Un autre filtre est couramment utilisé, le filtre **--type=service**. Nous pouvons indiquer à `systemctl` d'afficher uniquement le type d'unités qui nous intéresse. Par exemple, pour consulter uniquement les unités de service actives, nous pouvons utiliser :

```
[manal@fedora ~]$ systemctl list-units --type=service
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
abrt-journal-core.service	loaded	active	running	ABRT coredumpctl message creator
abrt-oops.service	loaded	active	running	ABRT kernel log watcher
abrt-xorg.service	loaded	active	running	ABRT Xorg log watcher
abrttd.service	loaded	active	running	ABRT Daemon
accounts-daemon.service	loaded	active	running	Accounts Service
alsa-state.service	loaded	active	running	Manage Sound Card State (restore and store)
auditd.service	loaded	active	running	Security Auditing Service
avahi-daemon.service	loaded	active	running	Avahi mDNS/DNS-SD Stack
chronyd.service	loaded	active	running	NTP client/server
colord.service	loaded	active	running	Manage, Install and Generate Color Profiles
cups.service	loaded	active	running	CUPS Scheduler
dbus-:1.15-org.freedesktop.problems@0.service	loaded	active	running	dbus-:1.15-org.freedesktop.problems@0.service
dbus-broker.service	loaded	active	running	D-Bus System Message Bus
dracut-shutdown.service	loaded	active	exited	Restore /run/initramfs on shutdown
firewalld.service	loaded	active	running	firewalld - dynamic firewall daemon
gdm.service	loaded	active	running	GNOME Display Manager
geoclue.service	loaded	active	running	Location Lookup Service
gssproxy.service	loaded	active	running	GSSAPI Proxy Daemon
kmod-static-nodes.service	loaded	active	exited	Create List of Static Device Nodes
low-memory-monitor.service	loaded	active	running	Low Memory Monitor
lvm2-monitor.service	loaded	active	exited	Monitoring of LVM2 mirrors, snapshots etc. using dmev
mcelog.service	loaded	active	running	Machine Check Exception Logging Daemon
ModemManager.service	loaded	active	running	Modem Manager
NetworkManager-dispatcher.service	loaded	active	running	Network Manager Script Dispatcher Service
NetworkManager-wait-online.service	loaded	active	exited	Network Manager Wait Online
NetworkManager.service	loaded	active	running	Network Manager



# Présentation générale d'états de système

## Liste de tous les fichiers de l'unité:

La commande `list-units` affiche uniquement les unités que `systemd` a tentées d'analyser et de charger en mémoire. Étant donné que `systemd` lira uniquement les unités dont il pense avoir besoin, les unités disponibles sur le système ne seront pas nécessairement toutes répertoriées. Pour voir tous les fichiers de l'unité disponibles au sein des chemins de `systemd`, notamment ceux que `systemd` n'a pas tenté de charger, vous pouvez utiliser la commande `list-unit-files` à la place :

```
[manal@fedora ~]$ systemctl list-unit-files
UNIT FILE                                STATE                                PRESET
UNIT FILE                                STATE                                PRESET
proc-sys-fs-binfmt_misc.automount       static                               -
-.mount                                  generated                           -
boot.mount                               generated                           -
dev-hugepages.mount                     static                               -
dev-mqueue.mount                        static                               -
home.mount                               generated                           -
proc-fs-nfsd.mount                      static                               -
proc-sys-fs-binfmt_misc.mount            disabled                            disabled
run-vmblock\x2dfuse.mount                disabled                            disabled
sys-fs-fuse-connections.mount            static                               -
sys-kernel-config.mount                  static                               -
sys-kernel-debug.mount                   static                               -
sys-kernel-tracing.mount                 static                               -
tmp.mount                                static                               -
var-lib-machines.mount                  static                               -
var-lib-nfs-rpc pipefs.mount             static                               -
cups.path                                enabled                             enabled
ostree-finalize-staged.path               disabled                            enabled
systemd-ask-password-console.path         static                               -
systemd-ask-password-plymouth.path         static                               -
systemd-ask-password-wall.path            static                               -
session-2.scope                          transient                           -
session-cl.scope                         transient                           -
abrt-journal-core.service                 enabled                             enabled
abrt-oops.service                        enabled                             enabled
abrt-pstoreoops.service                   disabled                            disabled
abrt-vmcore.service                      enabled                             enabled
abrt-xorg.service                        enabled                             enabled
```



# Présentation générale d'états de système

---

L'état indiquera généralement enabled, disabled, static ou masked. Dans ce contexte, static signifie que le fichier de l'unité ne contient pas de section install, qui sert à activer une unité. De ce fait, ces unités ne peuvent pas être activées. Habituellement, cela signifie que soit l'unité effectue une action unique, soit elle est uniquement utilisée qu'en tant que dépendance d'une autre unité et ne doit pas être exécutée seule.





# Gestion d'unité

Jusque-là, nous avons travaillé avec les services et affiché des informations sur l'unité et sur les fichiers de l'unité dont systemd a connaissance. Cependant, en utilisant d'autres commandes, nous pouvons trouver des informations plus spécifiques sur les unités.

## 1-Affichage du fichier de l'unité:

Pour afficher le fichier de l'unité que systemd a chargé sur son système, vous pouvez utiliser la commande **cat** qui a été ajoutée dans la version 209 de systemd).

```
[manal@fedora ~]$ systemctl cat sshd.service
# /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.target
Wants=sshd-keygen.target
# Migration for Fedora 38 change to remove group ownership for standard host keys
# See https://fedoraproject.org/wiki/Changes/SSHKeySignSuidBit
Wants=ssh-host-keys-migration.service

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/ssh
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target

# /usr/lib/systemd/system/service.d/10-timeout-abort.conf
# This file is part of the systemd package.
# See https://fedoraproject.org/wiki/Changes/Shorter_Shutdown_Timer.
#
# To facilitate debugging when a service fails to stop cleanly,
# TimeoutStopFailureMode=abort is set to "crash" services that fail to stop in
# the time allotted. This will cause the service to be terminated with SIGABRT
# and a coredump to be generated.
```

Activate Windows  
Go to Settings to activate Windows.



# Gestion d'unité

## 2-Affichage des dépendances:

Pour voir une arborescence des dépendances de l'unité, vous pouvez utiliser la commande **list-dependencies**.

```
[manal@fedora ~]$ systemctl list-dependencies sshd.service
sshd.service
● ssh-host-keys-migration.service
● system.slice
● sshd-keygen.target
○ sshd-keygen@ecdsa.service
○ sshd-keygen@ed25519.service
○ sshd-keygen@rsa.service
● sysinit.target
● dev-hugepages.mount
● dev-mqueue.mount
● dracut-shutdown.service
○ iscsi-onboot.service
○ iscsi-starter.service
● kmod-static-nodes.service
○ ldconfig.service
● lvm2-lvmpolld.socket
● lvm2-monitor.service
● plymouth-read-write.service
● plymouth-start.service
● proc-sys-fs-binfmt_misc.automount
○ selinux-autorelabel-mark.service
● sys-fs-fuse-connections.mount
● sys-kernel-config.mount
● sys-kernel-debug.mount
● sys-kernel-tracing.mount
○ systemd-ask-password-console.path
● systemd-binfmt.service
○ systemd-boot-random-seed.service
○ systemd-firstboot.service
○ systemd-hwdb-update.service
○ systemd-journal-catalog-update.service
```

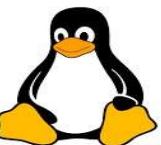


# Gestion d'unité

## 3-Vérification des propriétés de l'unité:

Pour consulter les propriétés de niveau inférieur d'une unité, vous pouvez utiliser la commande **show**.

```
[manal@fedora ~]$ systemctl show sshd.service
Type=notify
ExitType=main
Restart=on-failure
NotifyAccess=main
RestartUSec=42s
TimeoutStartUSec=45s
TimeoutStopUSec=45s
TimeoutAbortUSec=45s
TimeoutStartFailureMode=terminate
TimeoutStopFailureMode=abort
RuntimeMaxUSec=infinity
RuntimeRandomizedExtraUSec=0
WatchdogUSec=0
WatchdogTimestampMonotonic=0
RootDirectoryStartOnly=no
RemainAfterExit=no
GuessMainPID=yes
MainPID=5184
ControlPID=0
FileDescriptorStoreMax=0
NFileDescriptorStore=0
StatusErrno=0
Result=success
ReloadResult=success
CleanResult=success
UID=[not set]
GID=[not set]
NRestarts=0
OOMPolicy=stop
ReloadSignal=1
```



# Gestion d'unité

Pour afficher une seule propriété, vous pouvez faire utiliser la balise **-p** accompagnée du nom de la propriété.

Par exemple, pour voir les conflits que l'unité `sshd.service` a, vous pouvez saisir :

```
[manal@fedora ~]$ systemctl show sshd.service -p Conflicts
Conflicts=shutdown.target
[manal@fedora ~]$
```

## 4-Masquage et affichage des unités:

Dans la section Gestion de service, nous avons vu de quelle manière arrêter ou désactiver un service, mais `systemd` a également la possibilité de marquer une unité comme étant totalement impossible à démarrer, automatiquement ou manuellement, en la reliant à `/dev/null`. On dit alors que l'on « masque » l'unité, et il est possible de le faire avec la commande **mask**

```
[manal@fedora ~]$ sudo systemctl mask sshd.service
[sudo] Mot de passe de manal :
Created symlink /etc/systemd/system/sshd.service → /dev/null.
[manal@fedora ~]$
```



# Gestion d'unité

Tant qu'elle est masquée, tout démarrage automatique ou manuel du service sshd est impossible. Si vous vérifiez la list-unit-files, vous verrez que le service est maintenant répertorié comme étant masqué :

```
[manal@fedora ~]$ systemctl list-unit-files
```

```
sshd-keygen.service masked disabled  
sshd.service masked disabled
```

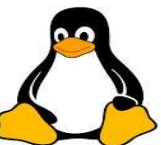
Si vous tentez de lancer le service, vous verrez s'afficher le message suivant :

```
[manal@fedora ~]$ sudo systemctl start sshd.service  
[sudo] Mot de passe de manal :  
Failed to start sshd.service: Unit sshd.service is masked.  
[manal@fedora ~]$
```

Pour afficher une unité et rendre son utilisation à nouveau possible, utilisez la commande **unmask**

```
[manal@fedora ~]$ sudo systemctl unmask sshd.service  
Removed "/etc/systemd/system/sshd.service".  
[manal@fedora ~]$
```

Cela renverra l'unité à l'état précédent, ce qui lui permettra son démarrage ou son activation.



# Les journaux

## Définition:

Les journaux sont un moyen important pour les démons de communiquer des informations à l'administrateur système. Ils peuvent contenir des informations sur les événements qui se produisent, les erreurs qui se produisent ou les performances du démon.

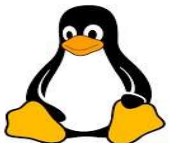
## Types de journaux:

**Il existe deux types principaux de journaux dans Linux :**

les journaux système et les journaux de démons.

**Les journaux système** sont créés par le noyau Linux et contiennent des informations sur les événements qui se produisent au niveau du système, tels que le démarrage et l'arrêt du système, l'accès aux périphériques ou les erreurs du noyau.

**Les journaux de démons** sont créés par les démons eux-mêmes et contiennent des informations sur les événements qui se produisent dans le cadre du fonctionnement du démon.



# Les journaux

## Les journaux système:

Les commandes de journaux système sont utilisées pour lire, analyser et gérer les journaux système. Les journaux système contiennent des informations sur les événements qui se produisent au niveau du système, tels que le démarrage et l'arrêt du système, l'accès aux périphériques ou les erreurs du noyau.

**voici quelques exemples de l'utilisation des commandes de journaux système :**

**Pour afficher tous les journaux système :**

en utilise la commande

**journalctl**

**journalctl** : C'est la commande principale pour interagir avec le journal système. Elle permet d'afficher et de gérer les journaux





# Les journaux

la commande **journalctl** : affiche l'intégralité du journal système depuis le démarrage actuel jusqu'au moment présent

```
meryeme@fedora:~ — journalctl
Oct 26 09:05:23 fedora kernel: Linux version 6.2.9-300.fc38.x86_64 (mockbuild@38f30b3c0c69453fae61718fc43f33bc) (gcc (G
Oct 26 09:05:23 fedora kernel: Command line: BOOT_IMAGE=(hd0,gpt2)/vmlinuz-6.2.9-300.fc38.x86_64 root=UUID=f8041c3a-afe
Oct 26 09:05:23 fedora kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Oct 26 09:05:23 fedora kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
Oct 26 09:05:23 fedora kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
Oct 26 09:05:23 fedora kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
Oct 26 09:05:23 fedora kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.
Oct 26 09:05:23 fedora kernel: signal: max sigframe size: 1776
Oct 26 09:05:23 fedora kernel: BIOS-provided physical RAM map:
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbff] usable
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x0000000000009fc00-0x0000000000009ffff] reserved
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x000000000000f0000-0x000000000000fffff] reserved
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x00000000000100000-0x00000000000dfffff] usable
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x00000000000dfff0000-0x00000000000dfffffff] ACPI data
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x000000000fec00000-0x000000000fec00fff] reserved
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x000000000fee00000-0x000000000fee00fff] reserved
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x000000000fff00000-0x000000000fffffff] reserved
Oct 26 09:05:23 fedora kernel: BIOS-e820: [mem 0x00000000100000000-0x0000000010cbfffff] usable
Oct 26 09:05:23 fedora kernel: NX (Execute Disable) protection: active
```

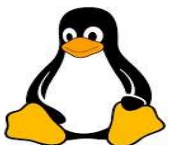




# Les journaux

La commande **journalctl -r** est utilisée pour afficher les journaux système dans l'ordre inverse

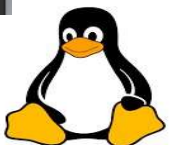
```
meryeme@fedora:~ — journalctl -r
Nov 20 13:18:20 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:20 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346746) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346591) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346559) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346528) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346497) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346405) is greater than comparison timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: W0 appears to be one of the offending windows with a timestamp>
Nov 20 13:18:19 fedora gnome-shell[1602]: Window manager warning: last_user_time (30346374) is greater than comparison timestamp>
```



# Les journaux

**La commande `journalctl -b`** : Afficher les messages du journal depuis le dernier démarrage

```
meryeme@fedora:~ — journalctl -b
Nov 21 06:37:51 fedora kernel: Kernel command line: BOOT_IMAGE=(hd0,gpt2)/vmlinuz-6.2.9-300.fc38.x86_64 root=UUID=f8041>
Nov 21 06:37:51 fedora kernel: Unknown kernel command line parameters "rhgb BOOT_IMAGE=(hd0,gpt2)/vmlinuz-6.2.9-300.fc3>
Nov 21 06:37:51 fedora kernel: random: crng init done
Nov 21 06:37:51 fedora kernel: Dentry cache hash table entries: 524288 (order: 10, 4194304 bytes, linear)
Nov 21 06:37:51 fedora kernel: Inode-cache hash table entries: 262144 (order: 9, 2097152 bytes, linear)
Nov 21 06:37:51 fedora kernel: mem auto-init: stack:all(zero), heap alloc:off, heap free:off
Nov 21 06:37:51 fedora kernel: software IO TLB: area num 1.
Nov 21 06:37:51 fedora kernel: Memory: 3643156K/3878456K available (18432K kernel code, 3223K rwddata, 14020K rodata, 43>
Nov 21 06:37:51 fedora kernel: SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Nov 21 06:37:51 fedora kernel: Kernel/User page tables isolation: enabled
Nov 21 06:37:51 fedora kernel: ftrace: allocating 51513 entries in 202 pages
Nov 21 06:37:51 fedora kernel: ftrace: allocated 202 pages with 4 groups
```



# Les journaux

**journalctl -f**: Afficher les messages en temps réel (suivi des nouveaux messages)

**Journalctl -x** : Afficher les messages avec plus de détails (informations supplémentaires)

## Les journaux de démons:

Les "journaux de démon" ou "logs de démon" font référence aux fichiers journaux qui enregistrent les activités, les erreurs, les événements et autres informations liées à un processus ou à un service spécifique sur un système d'exploitation.

Pour afficher les messages d'un demon spécifique en utilisant la commande:

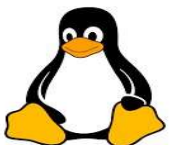
**journalctl -u <le nom de demon>**



# Les journaux

## Exemple de journal de demon sshd:

```
[manal@fedora ~]$ journalctl -u sshd
nov. 19 08:24:16 fedora systemd[1]: Starting sshd.service - OpenSSH server daemon...
nov. 19 08:24:16 fedora sshd[4476]: Server listening on 0.0.0.0 port 22.
nov. 19 08:24:16 fedora sshd[4476]: Server listening on :: port 22.
nov. 19 08:24:16 fedora systemd[1]: Started sshd.service - OpenSSH server daemon.
[manal@fedora ~]$ journalctl list-units --type=service
```



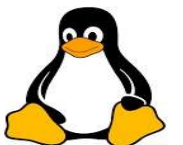
# La sécurité des démons

La sécurité des démons (services) sur un système Linux est un aspect cruciale pour maintenir l'intégrité et la confidentialité des systèmes.

**Voici quelques points importants concernant la sécurité des démons sous linux.**

## **Gestion des connexions réseau :**

- **Firewall** : Utilisez des pare-feu pour limiter les ports auxquels les démons ont accès depuis l'extérieur. Cela réduit la surface d'attaque potentielle.
- **Chiffrement des communications** : Si les démons gèrent des communications réseau, assurez-vous que ces communications sont chiffrées (TLS,SSH, etc.) pour éviter l'interception de données sensibles.



# La sécurité des démons

---

## Gestion des privilèges:

**Utilisateur spécifique:** Il est recommandé d'exécuter les démons avec un utilisateur spécifique et des privilèges restreints pour minimiser les risques.

**Droits d'accès aux fichiers :** Limitez les droits d'accès des démons aux fichiers et répertoires pour réduire les risques de compromission du système.

## Sécurité du réseau :

- **Chiffrement fort :** Utilisez des protocoles de chiffrement robustes pour sécuriser les communications entre les démons et d'autres systèmes.





# Les ressources

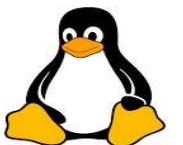
---

<https://www.malekal.com/systemd-service-linux-configuration-et-fonctionnement-daemon/>

<https://openclassrooms.com/fr/courses/1733551-gerez-votre-serveur-linux-et-ses-services/5236021-decouvrez-les-services-sous-linux>

<https://www.malekal.com/comment-utiliser-journalctl-pour-voir-lire-journaux-linux-systemd/>

<https://www.malekal.com/systemctl-linux--utilisation-et-exemples/>



MERCI POUR VOTRE ATTENTION