Administration système

Chapitre 5 - Processus, services et shell

Gorgoumack SAMBE

Université Assane Seck de Ziguinchor

Version 1.0 [1]

1. Novembre 2022

Objectifs

être capable de :

1. distinguer les principes de gestion des processus et services par le système Linux ;
2. manipuler/gérer les processus et services ;
3. distinguer et exploiter les fonctionnalités offertes par le shell ;

Plan

Plan

Processus et services

Processus

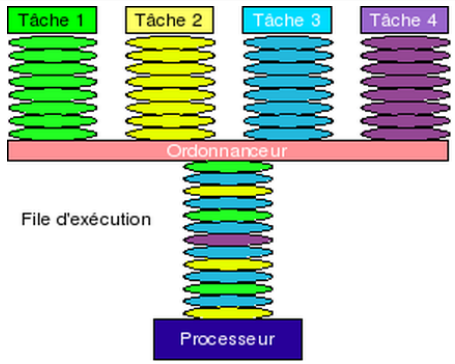
- Instance d'exécution d'un programme.
  1. Chargement en mémoire vive (RAM)
  2. Exécution par le processeur
- Identifié par un Process Identifier (PID)

Service

Processus non-associé à un terminal et qui tourne en arrière-plan
Exemples : interface graphique (Xorg), serveur de messagerie, serveur d'impression, ...

Système multitâche

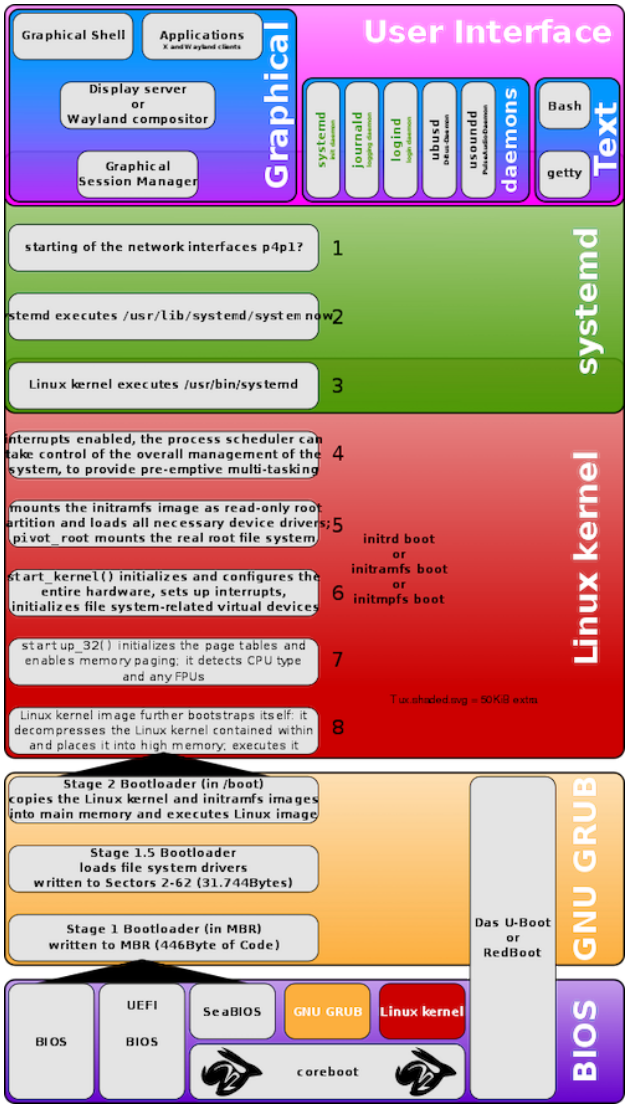
- Allocation du processeur par **quantum de temps** basé sur des **règles de priorités**.



Source : http ://si1.lmdsio.fr/lessons/systeme-exploitation

Démarrage du système

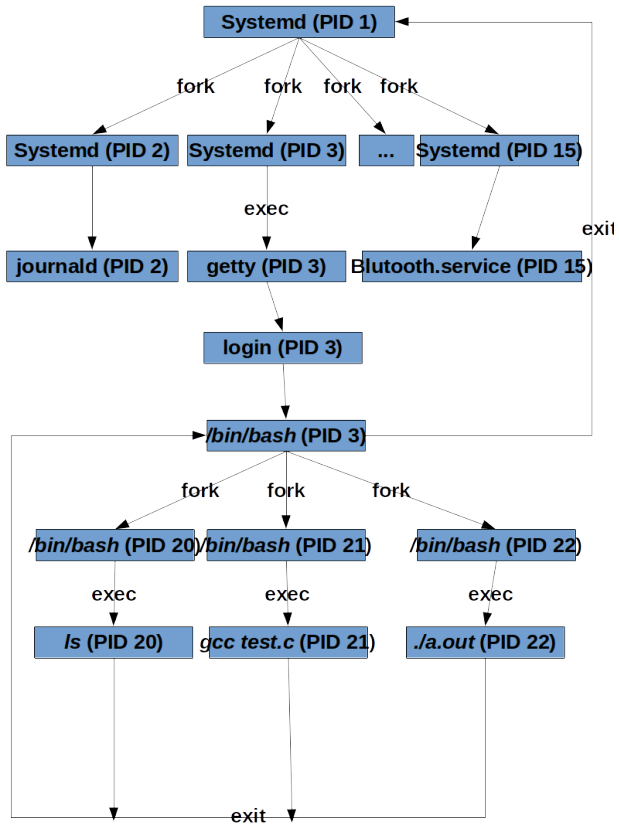
1. **BIOS/UEFI** : initialisation composants, détection partition d'amorçage ;
2. **Chargeur d'amorçage** : démarrage de plusieurs systèmes ;
3. **Mode noyau** : initialisation, démarrage systemd (1) ;
4. **systemd** : démarrage des services essentiels ;
5. **services** : : montage des périphèriques, services de messagerie, . . . , interface graphique ;
6. **utilisation du système** : connexion obligatoire.



Source : https ://commons.wikimedia.org/wiki/File :Linux_startup_process_wip.svg

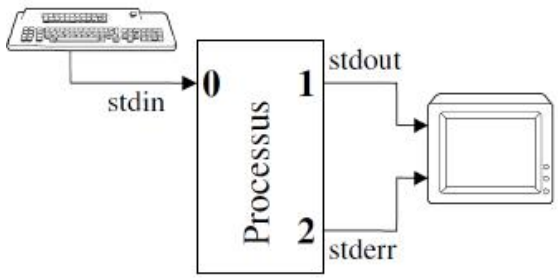
Processus, sécurité et confidentialité

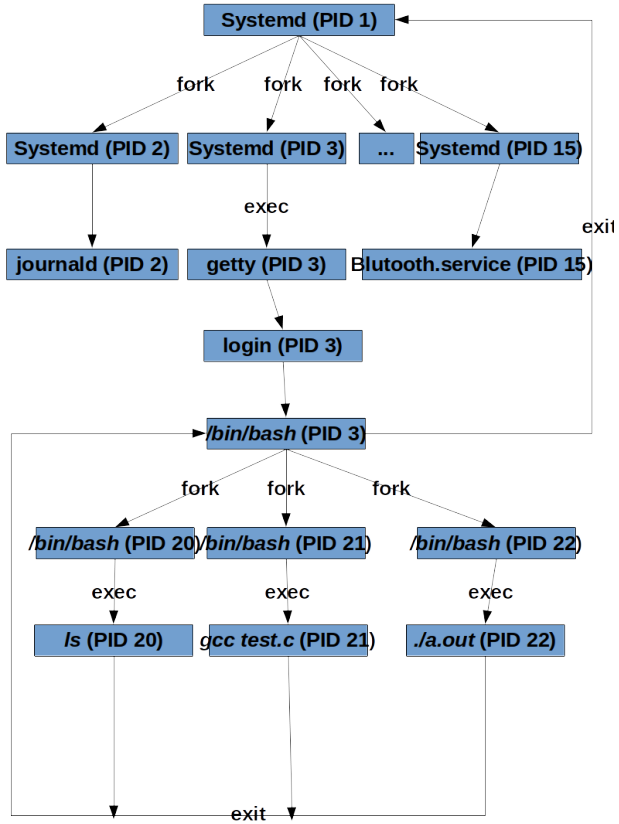
- **Création** : clonage par appel système (fork) ⇒ Arborescence ;



Processus, sécurité et confidentialité

- **Création** : clonage par appel système (fork) ⇒ Arborescence ;
- Processus ∈ **propriétaire** (qui l'a lancé) ;
  1. **zone mémoire dédiée** ;
  2. référencement de **fichiers** (inodes)
     - répertoire personnel (HD), répertoire de travail (PWD), **entrée standard** (0), **sortie standard** (1), **sortie d'erreur** (2).





Communication et ordonnancement

- **Communication**
  1. **signaux** :
     - généré par interruption clavier, erreur, ...
     - noyau, administrateur $\Longrightarrow$ tous les processus
     - processus régulier $\Longrightarrow$ même uid, gid ou groupe.
  2. **tubes**
     - Relie la sortie d'un processus à l'entrée d'un autre processus.
- **Ordonnancement**
  1. processus **temps réels**, police de priorité $\in$ [1,99]
     politique **round robin** (SCHED_RR) ou **FIFO** (SCHED_FIFO)
     dédié au **root**
  2. processus **normaux**, police de priorité=0
     politique du **temps partagé** (SCHED_OTHER)
     - priorité dynamique (nice=0) $\in$ [-20,19]

Plan

Processus

- **Affichage des processus actifs**
  - **ps** : cliché instantané
    Options : -e : tous les processus ; -f :affichage détaillé.
    $ ps -ef

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	Dec 6	?	1 :02	init
...							
olivier	321	319	0	10 :30 :34	ttyp1	0 :02	bash
olivier	324	321	0	10 :32 :12	ttyp1	0 :00	ps -ef

  - **top** : cliché dynamique, **pstree** : arborescence, **pgrep** : recherche.
- **Envoi de signal : kill, pkill**
  1. **2 - SIGINT** :Interruption ;
  2. **9 - SIGKILL - [CTRL] + [C]** : Terminaison immédiate ;
  3. **15 - SIGTERM** :Terminaison propre ;
  4. **18 - SIGCONT** : Reprise ;
  5. **19 - SIGSTOP- [CTRL] + [Z]** : Suspension.

Priorité des processus et tâches

- Priorité
  nice : Fixer une priorité
  renice : Modifier une priorité

- **Tâche** : processus utilisateur asynchrone.
  1. **processus synchrone** : perte du shell
  2. **processus asynchrone** : pas de perte du shell,le processus est appelé **tâche** (**job**) avec un **numéro**, & pour lancer une commande en tâche de fond.
- **Commandes de gestion des tâches**
  jobs : affichage, **bg** : passage en arrière-plan, **fg** : passage en avant-plan

Services

1. **Affichage de services**
   - systemctl list-unit-files –type=service
     tous les services
   - systemctl list-units –type=service
     services actifs

2. **Manipulation de services**
   systemctl ACTION <Nom_du_service>.service
   1. **start** : démarrer le service
   2. **stop** : arrêter le service
   3. reload : recharger le service
   4. status : connaître l'état du service
   5. enable : activer un service
   6. ...

3. **Target** : ensemble de services cohérents.
   - démarrage du système par niveau (runlevel).

Plan

Fonctionnement du shell

1. gestion des métacaractères ;
2. substitution des noms de fichiers et répertoires ;
3. gestion des variables ;
4. redirection des entrées et des sorties ;
5. gestion des tubes(pipe) ;
6. soumission des commandes au noyau.

Métacaractères et jokers

- Caractère traité avant soumission de la commande.

Caractère	Sémantique
$	Contenu d'une variable
;	Séparateur de commande
( )	Groupement de commandes
‘ ‘ $( )	Substitution de commande
< >	Redirection des entrées/sorties
|	Tube (pipe)
∼	Chemin absolu du répertoire de travail de l'utilisateur
&	Exécution en arrière plan d'une commande
* ? [] [∧]	jokers de substitution des noms de fichier
\ "	Dé-spécialisation
" " ’ ’	Délimitation - déspécialisation

- Jokers : substitution de noms de fichiers/commande

Caractère	Sémantique
?	N'importe quel caractère
*	N'importe quelle suite de caractères
[]	N'importe quel caractère parmi ceux spécifiés.
[∧]	N'importe quel caractère n'apparaissant pas dans les crochets

Variables

- **Variables locales**
  - déclaration : var=val
  - accès au contenu : $val
- **Variables spéciales**
  - **$$** : Identifiant de processus(PID) du shell courant ;
  - **$!** : Identifiant(PID) de la dernière tâche (job) lancée en arrière plan ;
  - **$?** : Code retour de la dernière commande (succés=0)
- **Variables d'environnement**
  - **contexte** d'exécution d'un programme ;
  - **env** : affichage des variables ;
  - **Exemples** : USERNAME,LOGIN, HOME, PATH,TERM, SHELL, LANG.

Redirection et tube

- **Redirection**
  1. **Entrée standard** : <
     - commande < fichier
       La commande va lire ses données à partir du fichier
  2. **Sortie standard** : >, > >
     - commande > fichier :
       Si fichier n'existe pas il est créé. Sinon il est écrasé. Le résultat de la commande est mis dans fichier.
     - commande > > fichier :
       Si fichier n'existe pas il est créé. Sinon le résultat est ajouté à la fin de fichier (mode append).
  3. **Sortie d'erreur** : 2 >, 2 > >
     - commande 2 > fichier
       Les messages d'erreur de la commande sont mis dans fichier.
     - commande 2> > fichier :
       Les erreurs sont ajoutées à la fin de fichier (mode append).
- **Tube** : Mécanisme permettant de relier la sortie standard d'un processus à l'entrée standard d'un autre, sans création de fichier intermédiaire.
  Exemple : $ du | sort -rn