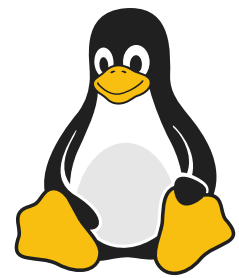
GNU/Linux

Processus et redirections

Nicolas Delanoue

Université d'Angers - Polytech Angers

POLYTECH°
ANGERS



Définition

Un processus représente une instance d'un programme en cours d'exécution.

Remarques

- Un processus n'est pas un programme ou une commande
- Un commande unique peut donner naissance à plusieurs processus.

Différents types de processus

Processus interactif Lancés par un utilisateur via une commande ou l'interface graphique, ex : `ls` ou bien `firefox`.

Processus automatisés Lancés automatiquement ou bien programmé depuis un terminal, ex : `cron, at`

Démon ou service Processus qui tourne continuement lancé principalement au démarrage, ex : serveur web, serveur `ssh`, ...

Processus léger Processus qui s'exécute sous l'autirité d'un processus principal via un développement multitache : les onglets de `firefox`,

Processus noyau Taches du noyau qui assurent le bon fonctionnement du système : ordonnanceur, accès au matériel, ...

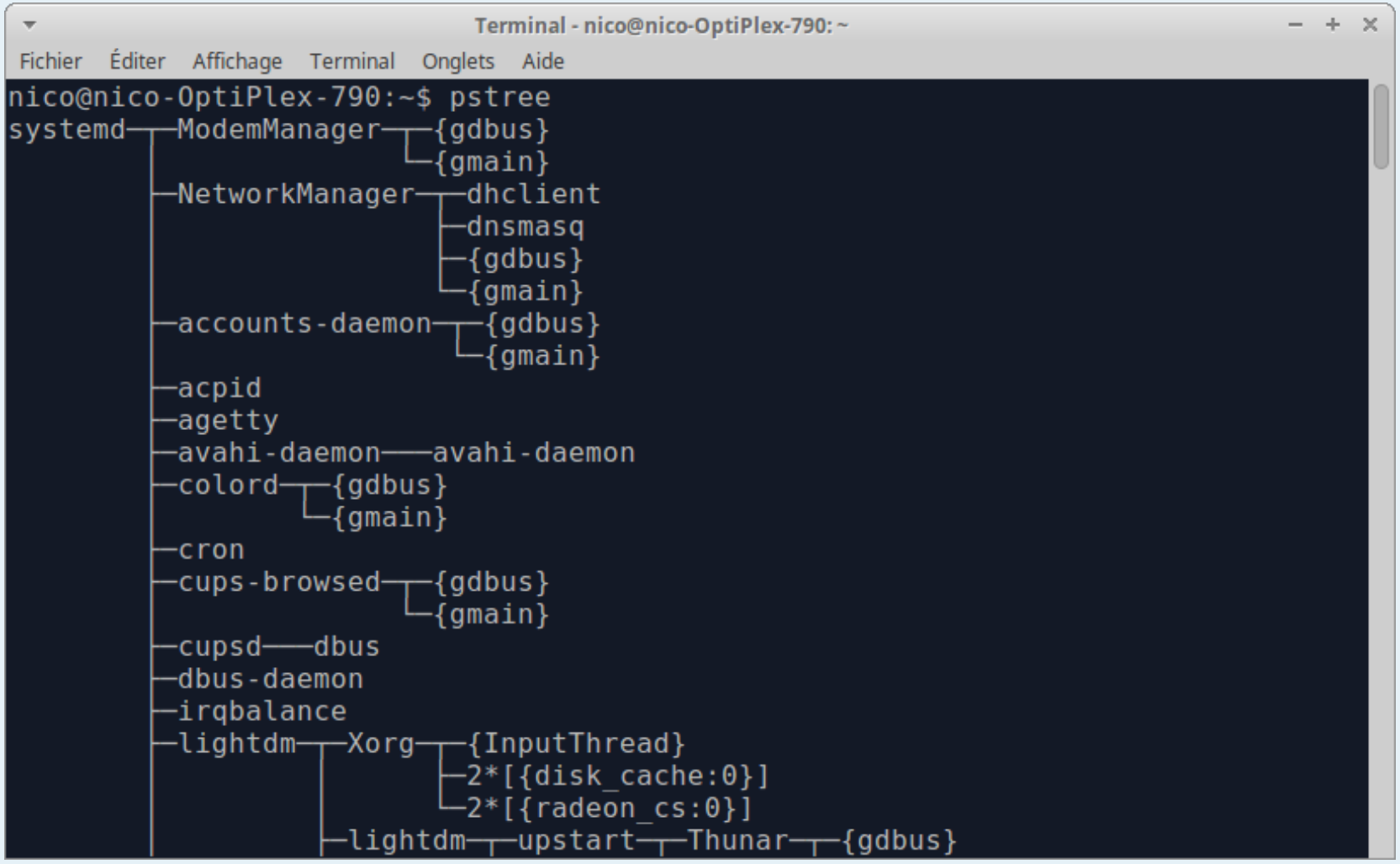
Caractéristiques d'un processus

- PID : un numéro d'identification
- PPID : le PID du processus parent
- utilisateur qui a lancé le processus
- sa priorité
- son état :
  - R (running),
  - S (sleeping),
  - T (stopped),
- place en mémoire, terminal, heure de lancement...

Quelques remarques :

- Chaque processus a un parent . . .
- tous, sauf `systemd` (le premier processus)
- Pour voir la hiérarchie des processus, on peut utiliser la commande `pstree`
- Note pour l'enseignement : démonstration de `ps aux`.

La commande pstree



Pour visualiser les processus en cours, on utilise la commande `ps`.

Terminal - nico@nico-OptiPlex-790: ~

Fichier Éditer Affichage Terminal Onglets Aide

```
nico@nico-OptiPlex-790:~$ ps
  PID TTY          TIME CMD
25343 pts/7    00:00:00 bash
25353 pts/7    00:00:00 ps
nico@nico-OptiPlex-790:~$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
nico     16674  0.0  0.0  22780  5496 pts/2    Ss+  nov.15   0:00 bash
nico     16775  0.0  0.0  22932  5768 pts/6    Ss+  nov.15   0:00 bash
nico     25343  0.6  0.0  22648  5216 pts/7    Ss   15:58   0:00 bash
nico     25354  0.0  0.0  37472  3360 pts/7    R+   15:58   0:00 ps u
nico@nico-OptiPlex-790:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0 185796  5964 ?        Ss   oct.26   0:07 /lib/systemd/s
root         2  0.0  0.0      0     0 ?        S    oct.26   0:00 [kthreadd]
root         4  0.0  0.0      0     0 ?        I<   oct.26   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        I<   oct.26   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0     0 ?        S    oct.26   0:01 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        I    oct.26  19:22 [rcu_sched]
root         9  0.0  0.0      0     0 ?        I    oct.26   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    oct.26   0:00 [migration/0]
root        11  0.0  0.0      0     0 ?        S    oct.26   0:01 [watchdog/0]
root        12  0.0  0.0      0     0 ?        S    oct.26   0:00 [cpuhp/0]
root        13  0.0  0.0      0     0 ?        S    oct.26   0:00 [cpuhp/1]
root        14  0.0  0.0      0     0 ?        S    oct.26   0:02 [watchdog/1]
```

Pour arrêter un processus (i.e. le tuer), il a 4 possibiltés :

- cliquant sur la croix en haut à droite sur le gui,

Pour arrêter un processus (i.e. le tuer), il a 4 possibiltés :

- cliquant sur la croix en haut à droite sur le gui,
- en lui demandant gentiment :
  `kill <PID>`
- sans lui demander son avis :
  `kill -9 <PID>`
- en indiquant son nom plutôt que son PID :
  `pkill <nomcommande>`

Remarque

Pour que ça marche, il faut être le propriétaire du processus (ou `root`).

Définition

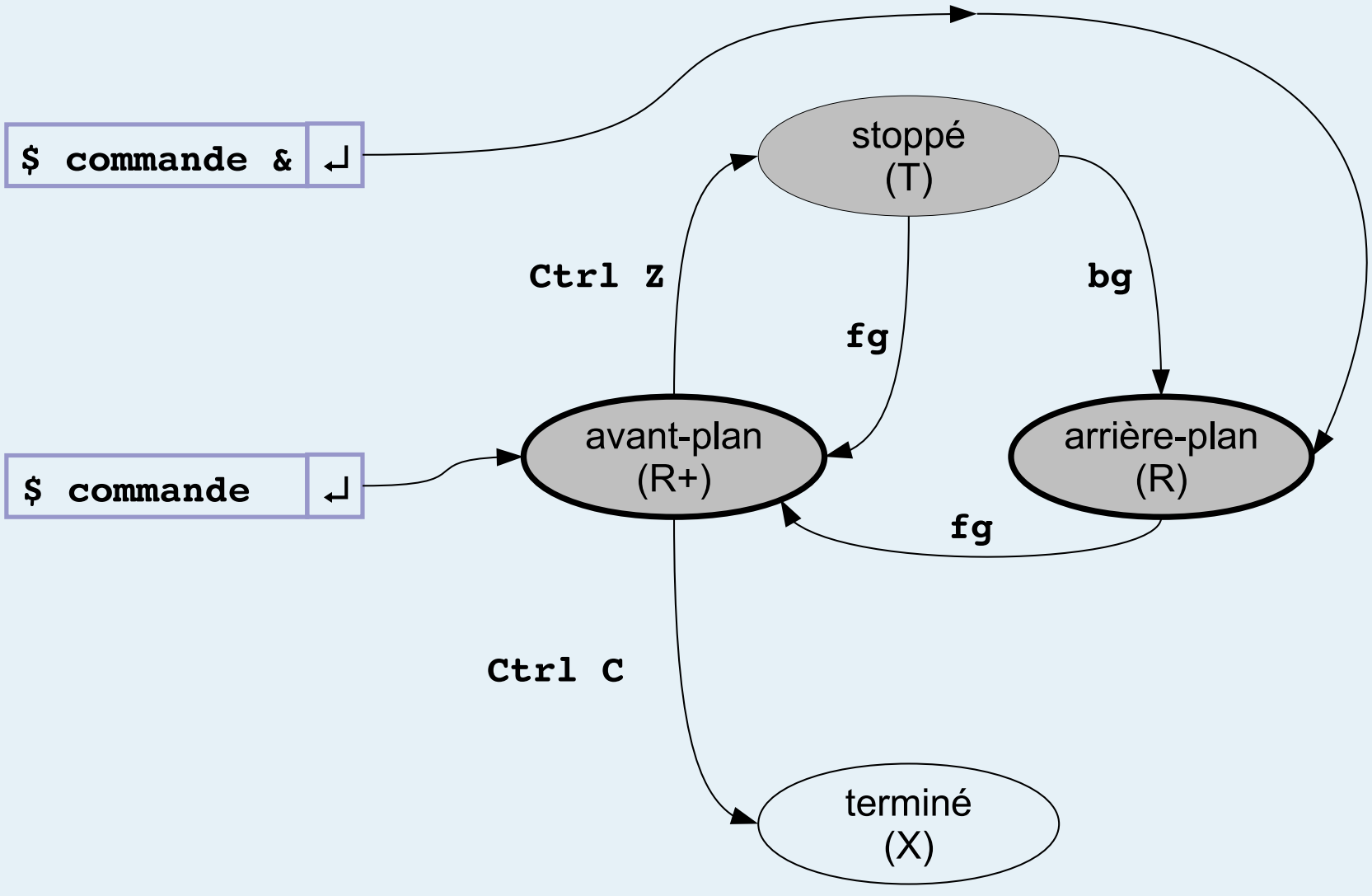
Quand on lance un processus dans une console, la console reste bloquée jusqu'à ce que le processus se termine. Le processus est en *avant-plan*.

Remarque

- On peut demander au processus de s'arrêter en tapant `Ctrl C`.
- Lancer le processus en arrière-plan pour continuer à utiliser la console en attendant :

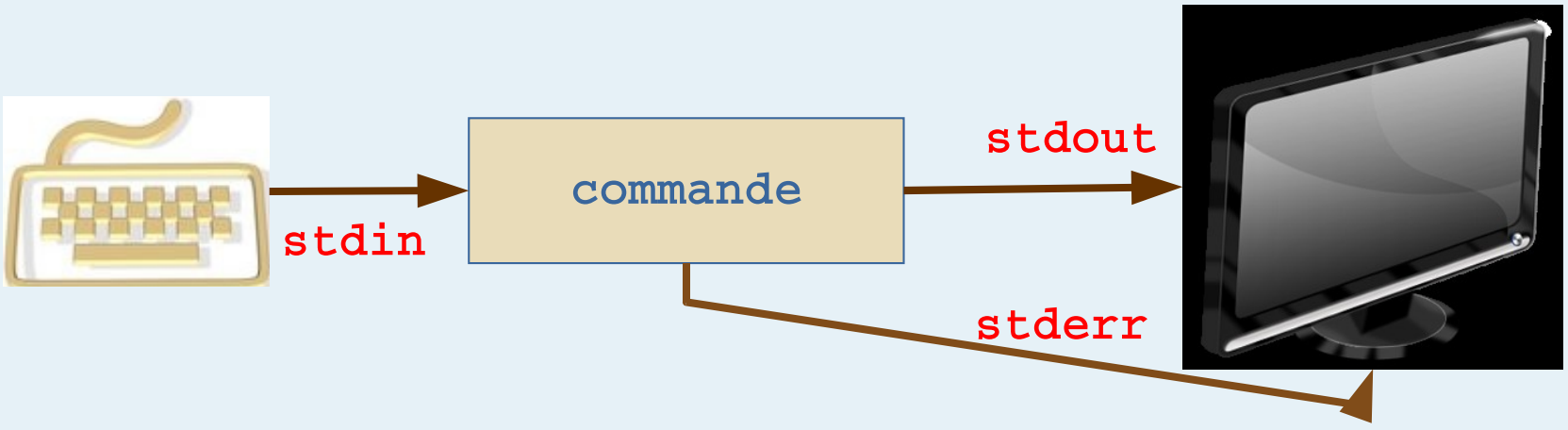
  `$<commande> &`
- Mettre en pause un processus lancé en avant-plan en tapant `Ctrl Z`.
- Pour le relancer
  - en arrière-plan : `bg`,
  - en avant-plan : `fg`.

Activités et pauses d'un processus



Une commande ouvre 3 flux ; par défaut :

- `stdin`, pour standard input,
- `stdout`, pour standard output,
- `stderr`, pour standard error.



Définition

Rediriger, c'est remplacer un de ces flux par un fichier ou par l'entrée/la sortie d'une autre commande.

Syntaxe des redirections

- < redirige l'entrée standard,
- > redirige la sortie standard,
- >> redirige la sortie standard (en ajout),
- 2> redirige la sortie d'erreur,
- &> redirige la sortie standard et la sortie d'erreur.

Exemple :

`ls > liste`	crée/écrase le fichier liste et y dirige la sortie de `ls`
`date >> fichier.txt`	ajoute à la fin du fichier `fichier.txt` la sortie de `date`
`wc -l < f2.txt`	envoie comme entrée à la commande `wc` le fichier `f2.txt`

Définition

Les tubes permettent de connecter deux commandes.

Exemple

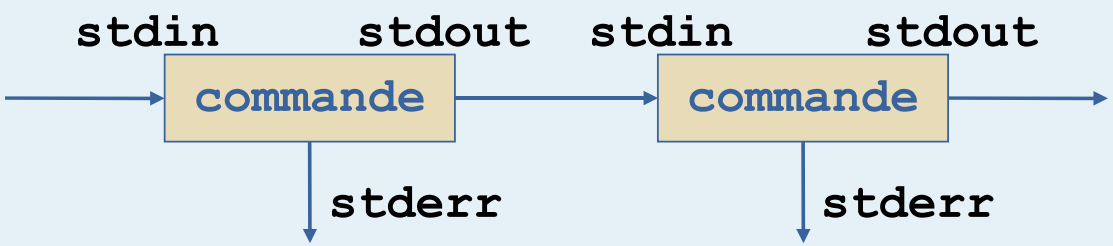
La suite de commandes

```
ls > temp.txt ; wc -l < temp ; rm temp
```

est équivalent à

```
ls | wc -l
```

Illustration



Les filtres

- `cat`
  - affiche le contenu des fichiers passés en paramètres (par défaut, stdin)
  - options `-b`, `-n`, `-v`.
- more
  - affiche page par page les fichiers passés en paramètres (par défaut, `stdin`)
  - `h` pour avoir le détail des commandes
- `less`
  - comme `more` mais retour en arrière possible
  - `q` pour quitter

Exemple

```
ps aux | more
```

Le filtre grep

- Le programme `grep` recherche, dans le fichier passé en paramètre, les lignes vérifiant une expression régulière donnée.
- syntaxe : `grep expr_reg [fichier]`

Exemple

- `grep 'toto' essai`
  cherche dans essai toutes les lignes qui contiennent le mot `toto`.
- `ls -l | grep network`
  cherche dans le résultat de `ls` toutes les lignes qui contiennent `network`.