

Les Signaux

Version 1

Pascal Malingrey

Académie Strasbourg

27 juin 2019

STOP ou ENCORE

Vous disposez dans votre dossier d'un fichier **ex1.py** (son contenu n'a pas d'importance pour le moment.) Dans un terminal vous exécuter la commande suivante :

```
# terminal  
| nsi@lin$ python3 ex1.py
```

1. Que constatez vous ?

STOP ou ENCORE

Vous disposez dans votre dossier d'un fichier **ex1.py** (son contenu n'a pas d'importance pour le moment.) Dans un terminal vous exécuter la commande suivante :

```
# terminal  
| nsi@lin$ python3 ex1.py
```

1. Que constatez vous ?
2. Comment interrompre le programme , tout en gardant le terminal actif ?

STOP ou ENCORE

Vous disposez dans votre dossier d'un fichier **ex1.py** (son contenu n'a pas d'importance pour le moment.) Dans un terminal vous exécuter la commande suivante :

```
# terminal  
| nsi@lin$ python3 ex1.py
```

1. Que constatez vous ?
2. Comment interrompre le programme , tout en gardant le terminal actif ?
3. On va appuyer sur une combinaison de touche Ctrl+C

Que s'est-il passé ?

La combinaison de touche `Ctrl+C` a interrompu l'exécution du programme. Cela signifie deux choses :

- ▶ "quelque chose" est à l'écoute du clavier
- ▶ le programme traite l'information de la combinaison `Ctrl+C`

Commande shell

La commande `cat` est utilisée qu'à titre d'exemple, sa fonctionnalité n'est pas importante ici. Par contre les suivantes sont utiles :

- ▶ `ps` : liste les processus dans le terminal
- ▶ `^z` : `Ctrl+z` suspend un processus
- ▶ `fg` : reprend un processus

Dans un terminal

Les commandes

terminal

```
nsi@lin$ cat
```

```
nsi@lin$ ^z #Appui Ctrl+z
```

```
nsi@lin$ ps
```

Dans un terminal

Les commandes

```
# terminal
nsi@lin$ cat
nsi@lin$ ^z #Appui Ctrl+z
nsi@lin$ ps
```

Le résultat

```
# terminal
PID TTY          TIME CMD
25280 pts/0        00:00:00 bash
26623 pts/0        00:00:00 cat
26624 pts/0        00:00:00 ps
```


Dans un terminal

Les commandes

```
# terminal
nsi@lin$ cat
nsi@lin$ ^z  #Appui Ctrl+z
nsi@lin$ ps
```

Le résultat

```
# terminal
PID TTY          TIME CMD
25280 pts/0        00:00:00 bash
26623 pts/0        00:00:00 cat
26624 pts/0        00:00:00 ps
```

On voit trois processus

- ▶ le bash (qui correspond à notre terminal)
- ▶ cat (qu'on a lancé)
- ▶ ps (qui liste nos processus)

Remarque : Vous constatez que les numéros de PID ne sont pas les mêmes chez vous.

Stoppons le processus cat

Nous allons envoyer un signal de terminaison (comparable au Ctrl-C) au processus **cat** par le biais de la commande **killall**. **killall** envoie un signal aux processus dont le nom est indiqué avec le numéro du signal.

Regardons ce qu'il en est des processus

Les commandes

```
# terminal
nsi@lin$ killall -2 cat
nsi@lin$ ps
```

Stoppons le processus cat

Nous allons envoyer un signal de terminaison (comparable au Ctrl-C) au processus **cat** par le biais de la commande **killall**. **killall** envoie un signal aux processus dont le nom est indiqué avec le numéro du signal.

Regardons ce qu'il en est des processus

Le résultat

Les commandes

```
# terminal
nsi@lin$ killall -2 cat
nsi@lin$ ps
```

```
# terminal
PID TTY          TIME CMD
25280 pts/0        00:00:00 bash
26623 pts/0        00:00:00 cat
26624 pts/0        00:00:00 ps
```

Aucune différence !!

Reprise du processus

On va demander une reprise du processus avec la commande `fg`

Les commandes

```
# terminal  
|  
nsi@lin$ fg  
nsi@lin$ ps
```

Reprise du processus

On va demander une reprise du processus avec la commande `fg`

Le résultat

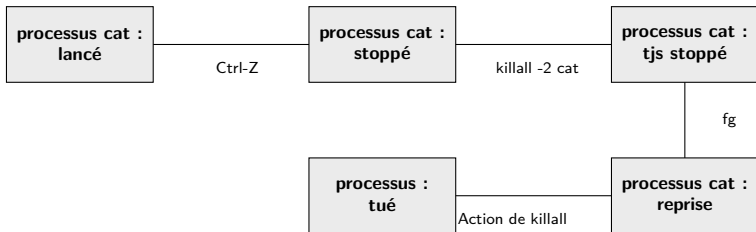
Les commandes

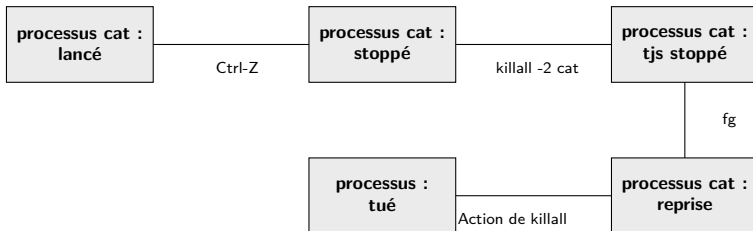
terminal

```
nsi@lin$ fg  
nsi@lin$ ps
```

terminal

PID	TTY	TIME	CMD
25280	pts/0	00:00:00	bash
26624	pts/0	00:00:00	ps





Conclusion

Suite à la reprise du processus, le signal **Ctrl-C** a été exécuté.

Le signal d'interruption n'est traité que lorsque le processus cat redevient actif, c'est donc bien le processus qui traite l'information du signal **2**, dont il est destinataire.

Cela signifie également que le signal **2** a été mémorisé par le système.

Définition d'un signal

Definition 1

Un signal est :

- ▶ un message envoyé par le noyau de manière *asynchrone* à :
 - ▶ un processus ; ou
 - ▶ un groupe de processus
- ▶ pour indiquer un événement système important

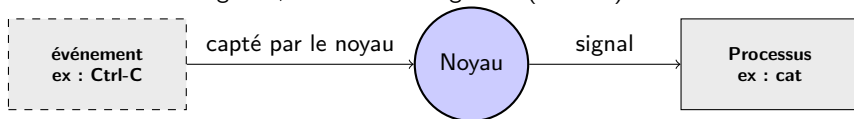
Définition d'un signal

Definition 1

Un signal est :

- ▶ un message envoyé par le noyau de manière *asynchrone* à :
 - ▶ un processus ; ou
 - ▶ un groupe de processus
- ▶ pour indiquer un événement système important

Le message peut être à l'initiative d'un autre processus. Cette communication limitée (**seul le numéro du signal est envoyé**) entre les processus. La norme POSIX définit un certain nombre de signaux, environ une vingtaine. (voir 3.3)



Quelques événements possibles

- ▶ frappe de caractère dans un terminal

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ `Ctrl+C`

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ Ctrl+C
 - ▶ Ctrl+Z etc.

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ Ctrl+C
 - ▶ Ctrl+Z etc.
- ▶ terminaison d'un processus

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ Ctrl+C
 - ▶ Ctrl+Z etc.
- ▶ terminaison d'un processus
- ▶ un problème matériel : division par zéro, problème d'adressage, défaillance d'alimentation électrique, etc.

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ Ctrl+C
 - ▶ Ctrl+Z etc.
- ▶ terminaison d'un processus
- ▶ un problème matériel : division par zéro, problème d'adressage, défaillance d'alimentation électrique, etc.
- ▶ l'expiration de délai préprogrammé (fonction `alarm()`)

Quelques événements possibles

- ▶ frappe de caractère dans un terminal
 - ▶ Ctrl+C
 - ▶ Ctrl+Z etc.
- ▶ terminaison d'un processus
- ▶ un problème matériel : division par zéro, problème d'adressage, défaillance d'alimentation électrique, etc.
- ▶ l'expiration de délai préprogrammé (fonction `alarm()`)
- ▶ ...

Lister les signaux

Exécuter la commande ci-dessous pour avoir la liste des signaux disponibles

```
# terminal  
| nsi@lin$ killall -l
```

La norme POSIX distingue 32 signaux dont les principaux sont :

Numéro	Nom	Signification	Comportement
1	SIGHUP	Hang-up (fin de connexion)	T(erminaison)
2	SIGINT	Interruption (Ctrl-C)	T
	SIGQUIT	Interruption forte (Ctrl-\)	T + core
	SIGFPE	Erreur arithmétique	T + core
9	SIGKILL	Interruption immédiate et absolue	T + core
	SIGSEGV	Violation des protections mémoire	T + core
	SIGPIPE	Écriture sur un pipe sans lecteurs	T
20	SIGTSTP	Arrêt temporaire(Ctrl-Z)	Suspension
18	SIGCONT	Redémarrage d'un fils arrêté	Ignoré
	SIGCHLD	un des fils est mort ou arrêté	Ignoré
14	SIGALRM	Interruption d'horloge	Ignoré
19	SIGSTOP	Arrêt temporaire	Suspension
	SIGUSR1	Émis par un processus utilisateur	T
	SIGUSR2	Émis par un processus utilisateur	T

T : terminaison du processus ; core : création d'un fichier d'image mémoire

Remarque

Quelle différence entre SIGSTP et SIGSTOP, de même entre SIGINT et SIGKILL ?

Certains signaux ne peuvent être bloqués ou ignorés par le processus, c'est le cas de SIGSTOP et SIGKILL.

- ▶ SIGINT/SIGSTP laisse la possibilité au processus d'ignorer et/ou de contrôler le signal
- ▶ SIGKILL/SIGSTOP aucun moyen de passer outre la demande d'interruption.

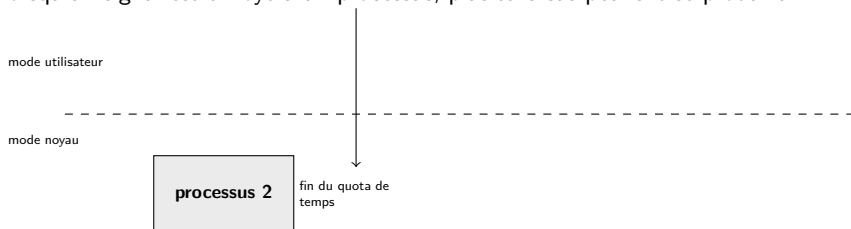
Pour la liste de toutes les actions des signaux :

<http://www.linux-france.org/article/man-fr/man7/signal-7.html>

Résumé

La prise en compte d'un signal (on parle de **délivrance**) ne peut avoir lieu que dans une circonstance bien particulière : la bascule du mode système au mode utilisateur.

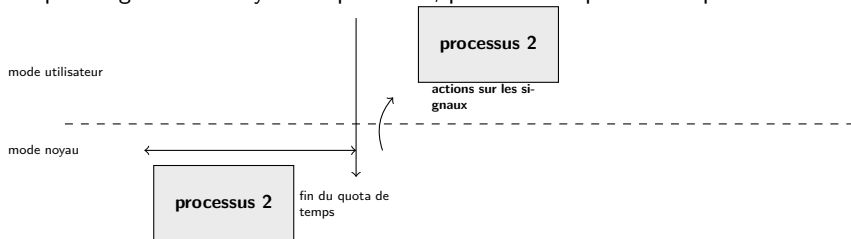
Lorsqu'un signal est envoyé à un processus, plusieurs cas peuvent se produire :



Résumé

La prise en compte d'un signal (on parle de **délivrance**) ne peut avoir lieu que dans une circonstance bien particulière : la bascule du mode système au mode utilisateur.

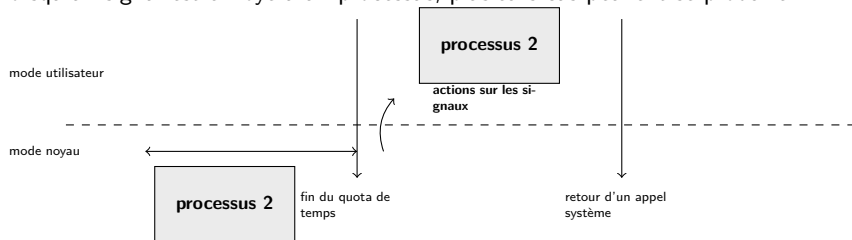
Lorsqu'un signal est envoyé à un processus, plusieurs cas peuvent se produire :



Résumé

La prise en compte d'un signal (on parle de **délivrance**) ne peut avoir lieu que dans une circonstance bien particulière : la bascule du mode système au mode utilisateur.

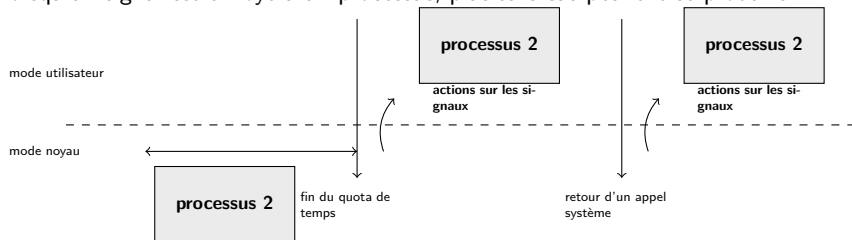
Lorsqu'un signal est envoyé à un processus, plusieurs cas peuvent se produire :



Résumé

La prise en compte d'un signal (on parle de **délivrance**) ne peut avoir lieu que dans une circonstance bien particulière : la bascule du mode système au mode utilisateur.

Lorsqu'un signal est envoyé à un processus, plusieurs cas peuvent se produire :



Résumé

- ▶ Le processus est en mode utilisateur. La délivrance devra alors attendre d'abord le passage du processus en mode noyau puis son retour au mode utilisateur. Pendant tout ce temps, le signal sera **pendant**, c'est à dire en attente de délivrance.

Résumé

- ▶ Le processus est en mode utilisateur. La délivrance devra alors attendre d'abord le passage du processus en mode noyau puis son retour au mode utilisateur. Pendant tout ce temps, le signal sera **pendant**, c'est à dire en attente de délivrance.
- ▶ Le processus est en mode noyau, par définition non interruptible. Le signal sera délivré dès que le processus reviendra au mode utilisateur : le signal est donc **pendant**. Lorsque le signal est délivré, la procédure qui lui est associée (son gestionnaire ou **handler**) est appelée. L'action du signal peut être :
 1. le comportement par défaut (par exemple l'interruption)
 2. l'ignorance
 3. le traitement personnalisé
 4. le masquage (blocage)

Etats d'un signal

Definition 2

Donc les différents états d'un signal sont :

généré/émis : L'événement associé au signal s'est produit

Etats d'un signal

Definition 2

Donc les différents états d'un signal sont :

généré/émis : L'événement associé au signal s'est produit

délivré : L'action associée au signal a été exécutée

Etats d'un signal

Definition 2

Donc les différents états d'un signal sont :

généré/émis : L'événement associé au signal s'est produit

délivré : L'action associée au signal a été exécutée

pendant : Le signal émis n'a pas encore été pris en compte

Etats d'un signal

Definition 2

Donc les différents états d'un signal sont :

généré/émis : L'événement associé au signal s'est produit

délivré : L'action associée au signal a été exécutée

pendant : Le signal émis n'a pas encore été pris en compte

bloqué/masqué : La prise en compte du signal est volontairement différée.

Compléments

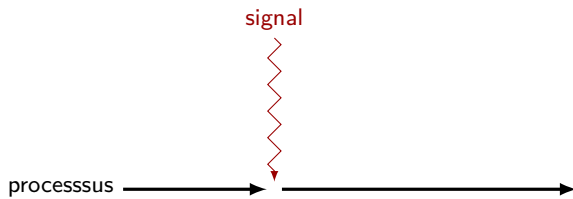
La structure de données interne (une par processus) gérant les signaux est un vecteur indexé sur les numéros de signaux et dont chaque case comporte 3 informations :

- ▶ Un **booléen** indiquant si le signal est pendant. Cette information est un booléen unique. Ce qui signifie que si un processus a déjà un signal d'un certain type pendant, il est inutile de lui envoyer à nouveau un signal du même type, celui-ci sera ignoré.
- ▶ Un **booléen** indiquant si les signaux de ce type sont bloqués.
- ▶ Un pointeur désignant le gestionnaire.

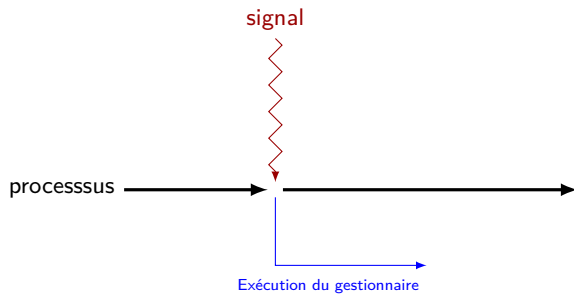
Le gestionnaire

processsus →

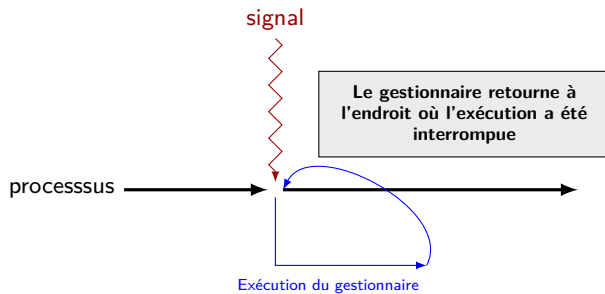
Le gestionnaire



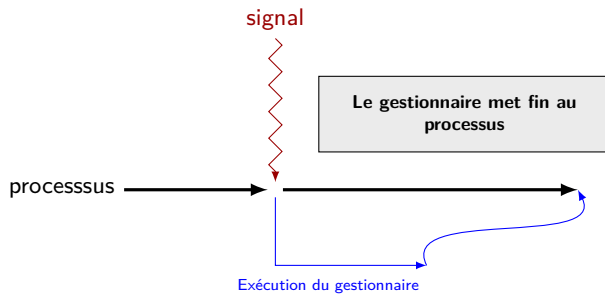
Le gestionnaire



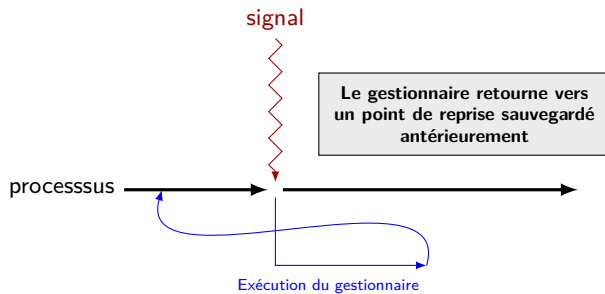
Le gestionnaire



Le gestionnaire



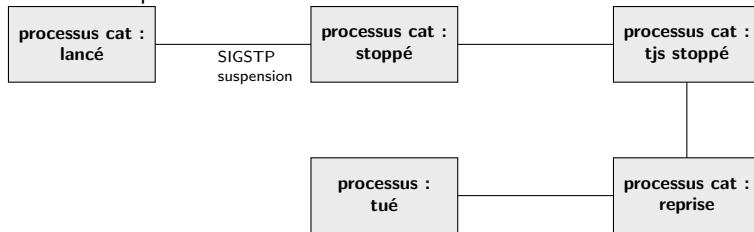
Le gestionnaire



Application

Reprenons l'approche 2.

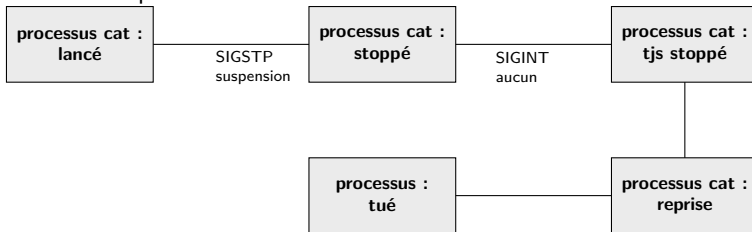
Indiquez le nom (dans la nomenclature POSIX) des signaux envoyés et les actions entre chacune des phases.



Application

Reprenons l'approche 2.

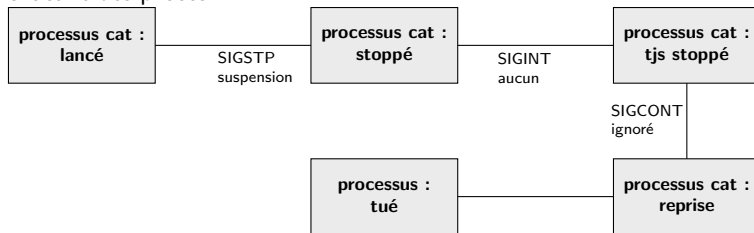
Indiquez le nom (dans la nomenclature POSIX) des signaux envoyés et les actions entre chacune des phases.



Application

Reprenons l'approche 2.

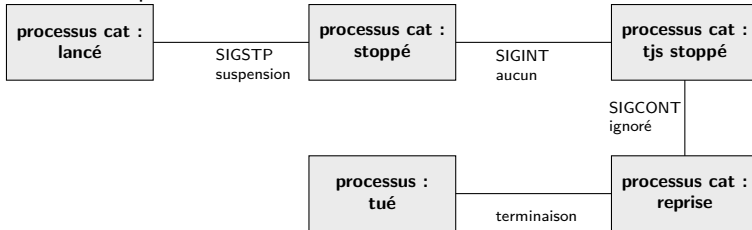
Indiquez le nom (dans la nomenclature POSIX) des signaux envoyés et les actions entre chacune des phases.



Application

Reprenons l'approche 2.

Indiquez le nom (dans la nomenclature POSIX) des signaux envoyés et les actions entre chacune des phases.



Retour à la programmation

Nous avons vu dans la première partie que certains signaux peuvent être interceptés par le processus.

Regardons comment faire à l'aide de Python



Les signaux avec Python

- **signal** : la librairie `signal` permet de travailler avec les signaux qui peuvent être modifiés (`SIGINT`, `SIGTERM`, `SIGSTP`, `SIGALRM`...)
- **`signal.signal(monsignal, mafonction)`** : permet d'exécuter la fonction **`mafonction`**, lors de l'apparition du signal **`monsignal`**. Il s'agit du gestionnaire (handler) du signal.
- **`signal.alarm(t)`** : envoie un signal `alarm` après un délai de t secondes.
- **`signal.SIG...`** : permet de faire référence au signal `SIG...` (en fait on récupère le numéro du signal).
par exemple `signal.SIGTERM` fait référence au `Ctrl-C`.

Exercice 1

Exercice 1

*Reprenez le fichier **ex1.py** et le modifier pour que le signal `SIGINT`(Ctrl-C) ne puisse pas interrompre le programme.*

Exercice 1

Exercice 1

Reprenez le fichier **ex1.py** et le modifier pour que le signal `SIGINT` (Ctrl-C) ne puisse pas interrompre le programme.

```
# ex1b.py

import signal
import time

def mongestionnaire(signum, stack):
    print("Ctrl-C est désactivé")

signal.signal(signal.SIGINT, mongestionnaire)

i = 1
while 1:
    print("itération :{}".format(i))
    time.sleep(1)
    i += 1
```

Exercice 2

Voici un programme :

```
import signal
import time

def mongestionnaire(signum, stack):
    print('Alarme :', time.ctime())

signal.signal(signal.SIGALRM, mongestionnaire)
signal.alarm(2)

print('Première:', time.ctime())
time.sleep(4)
print('Terminale :', time.ctime())
```

1. Expliquer les commandes des lignes 6 et 7.

Exercice 2

Voici un programme :

```
import signal
import time

def mongestionnaire(signum, stack):
    print('Alarme :', time.ctime())

signal.signal(signal.SIGALRM, mongestionnaire)
signal.alarm(2)

print('Première:', time.ctime())
time.sleep(4)
print('Terminale :', time.ctime())
```

1. Expliquer les commandes des lignes 6 et 7.

Ligne 6 : on associe au signal **SIGALRM**, la fonction **mongestionnaire** comme handler.

Ligne 7 : on envoie le signal **SIGALRM** au processus actuel avant un temps de 2 secondes d'attente.

2. Que va afficher le programme, si l'heure de début est 12 :00 :00 (12h) ?

Exercice 2

Voici un programme :

```
import signal
import time

def mongestionnaire(signum, stack):
    print('Alarme :', time.ctime())

signal.signal(signal.SIGALRM, mongestionnaire)
signal.alarm(2)

print('Première:', time.ctime())
time.sleep(4)
print('Terminale :', time.ctime())
```

1. Expliquer les commandes des lignes 6 et 7.

Ligne 6 : on associe au signal SIGALRM, la fonction mongestionnaire comme handler.

Ligne 7 : on envoie le signal SIGALRM au processus actuel avant un temps de 2 secondes d'attente.

2. Que va afficher le programme, si l'heure de début est 12 :00 :00 (12h) ?

Première : 12 :00 :00

Alarme : 12 :00 :02

Terminale : 12 :00 :04

Exercice 3

Notre programme possède une fonction **inconnue** qui peut prendre beaucoup de temps. Nous aimerions qu'au bout de 5s, celle-ci soit automatiquement interrompue. Apporter les modifications nécessaires, pour réaliser ce que l'on souhaite.

```
# La fonction inconnue
def inconnue(n):
    while n>0:
        print('je travaille encore ',n)
        time.sleep(1)
        n -= 1
```

un corrigé

```
# ex3.py
import signal, time, sys

def mongestionnaire(signum,stack):
    print ("Arrêt du programme")
    sys.exit(0)

def inconnue(n):
    signal.signal(signal.SIGALRM,mongestionnaire)
    signal.alarm(5)
    while n>0:
        print('je travaille',n)
        time.sleep(1)
        n -= 1
```


Exercice 4

*Un programme attend une réponse de l'utilisateur suite à une commande **input**. On aimerait qu'au bout de 5 secondes un message s'affiche, pour demander de répondre dans les 5 prochaines secondes et si tel n'est pas le cas le programme s'interrompt.*

#

ex4.py

```
import signal

TIMEOUT = 5 # délai de rigueur
CPT = 1

def mongestionnaire(signum,stack):
    global CPT
    if CPT == 1:
        print("Je suis généreux, je vous laisse encore 5 secondes")
        CPT += 1
        signal.alarm(TIMEOUT)
    else:
        print('Trop tard!')
        raise TimeoutError

signal.signal(signal.SIGALRM, mongestionnaire)

def input_delai():
    try:
        print ('Vous avez {} secondes pour répondre'.format(TIMEOUT))
        rep = input()
        return rep
    except TimeoutError: # temps dépassé
        return None

# le chrono tourne
signal.alarm(TIMEOUT)
s = input_delai()
# Ne pas oublier d'arrêter le chrono pour poursuivre
signal.alarm(0)
if s:
    print( 'votre réponse est:',s)
else:
```

Exercice 5

Nous avons deux programmes qui permettent de communiquer entre deux personnes (ex le jeu du juste prix). Le premier programme **serveur.py** va créer le canal de communication et le client **client.py** va pouvoir s'y connecter.

1. Lancez dans deux shells différents le serveur avec **python3 serveur.py**, puis **python3 client.py** et testez la communication.
2. À l'aide d'une copie d'écran avec trois shell bash (et uniquement trois sont lancés),

```

Terminal
Pascal Malingrey: Fichier Édition Affichage Recherche Terminal Aide
$ python3 serveur.py
pid du serveur 12000
pid du père du processus 12000
connection de: 1'127.0.0.1', 60876)

Terminal
Pascal Malingrey: Fichier Édition Affichage Recherche Terminal Aide
1 | | 0.75 | 5 | 0.85 |
2 | | 0.85 | 6 | | 0.75 |
3 | | 0.75 | 7 | | 0.85 |
4 | | 1.75 | 8 | | 0.85 |
root | | | | | 1.586704841 | Taux: 139, 518 thr: 1 running
$ cat /dev/null
1.586704841 | Taux: 139, 518 thr: 1 running
Last average: 0.51 0.16 0.19
uptime: 47:00:12

Terminal
Pascal Malingrey: Fichier Édition Affichage Recherche Terminal Aide
$ python3 client.py
12000
12007
12008
12009
12010
12011
12012
12013
12014
12015
12016
12017
12018
12019
12020
12021
12022
12023
12024
12025
12026
12027
12028
12029
12030
12031
12032
12033
12034
12035
12036
12037
12038
12039
12040
12041
12042
12043
12044
12045
12046
12047
12048
12049
12050
12051
12052
12053
12054
12055
12056
12057
12058
12059
12060
12061
12062
12063
12064
12065
12066
12067
12068
12069
12070
12071
12072
12073
12074
12075
12076
12077
12078
12079
12080
12081
12082
12083
12084
12085
12086
12087
12088
12089
12090
12091
12092
12093
12094
12095
12096
12097
12098
12099
12100
12101
12102
12103
12104
12105
12106
12107
12108
12109
12110
12111
12112
12113
12114
12115
12116
12117
12118
12119
12120
12121
12122
12123
12124
12125
12126
12127
12128
12129
12130
12131
12132
12133
12134
12135
12136
12137
12138
12139
12140
12141
12142
12143
12144
12145
12146
12147
12148
12149
12150
12151
12152
12153
12154
12155
12156
12157
12158
12159
12160
12161
12162
12163
12164
12165
12166
12167
12168
12169
12170
12171
12172
12173
12174
12175
12176
12177
12178
12179
12180
12181
12182
12183
12184
12185
12186
12187
12188
12189
12190
12191
12192
12193
12194
12195
12196
12197
12198
12199
12200
12201
12202
12203
12204
12205
12206
12207
12208
12209
12210
12211
12212
12213
12214
12215
12216
12217
12218
12219
12220
12221
12222
12223
12224
12225
12226
12227
12228
12229
12230
12231
12232
12233
12234
12235
12236
12237
12238
12239
12240
12241
12242
12243
12244
12245
12246
12247
12248
12249
12250
12251
12252
12253
12254
12255
12256
12257
12258
12259
12260
12261
12262
12263
12264
12265
12266
12267
12268
12269
12270
12271
12272
12273
12274
12275
12276
12277
12278
12279
12280
12281
12282
12283
12284
12285
12286
12287
12288
12289
12290
12291
12292
12293
12294
12295
12296
12297
12298
12299
12300
12301
12302
12303
12304
12305
12306
12307
12308
12309
12310
12311
12312
12313
12314
12315
12316
12317
12318
12319
12320
12321
12322
12323
12324
12325
12326
12327
12328
12329
12330
12331
12332
12333
12334
12335
12336
12337
12338
12339
12340
12341
12342
12343
12344
12345
12346
12347
12348
12349
12350
12351
12352
12353
12354
12355
12356
12357
12358
12359
12360
12361
12362
12363
12364
12365
12366
12367
12368
12369
12370
12371
12372
12373
12374
12375
12376
12377
12378
12379
12380
12381
12382
12383
12384
12385
12386
12387
12388
12389
12390
12391
12392
12393
12394
12395
12396
12397
12398
12399
12400
12401
12402
12403
12404
12405
12406
12407
12408
12409
12410
12411
12412
12413
12414
12415
12416
12417
12418
12419
12420
12421
12422
12423
12424
12425
12426
12427
12428
12429
12430
12431
12432
12433
12434
12435
12436
12437
12438
12439
12440
12441
12442
12443
12444
12445
12446
12447
12448
12449
12450
12451
12452
12453
12454
12455
12456
12457
12458
12459
12460
12461
12462
12463
12464
12465
12466
12467
12468
12469
12470
12471
12472
12473
12474
12475
12476
12477
12478
12479
12480
12481
12482
12483
12484
12485
12486
12487
12488
12489
12490
12491
12492
12493
12494
12495
12496
12497
12498
12499
12500
12501
12502
12503
12504
12505
12506
12507
12508
12509
12510
12511
12512
12513
12514
12515
12516
12517
12518
12519
12520
12521
12522
12523
12524
12525
12526
12527
12528
12529
12530
12531
12532
12533
12534
12535
12536
12537
12538
12539
12540
12541
12542
12543
12544
12545
12546
12547
12548
12549
12550
12551
12552
12553
12554
12555
12556
12557
12558
12559
12560
12561
12562
12563
12564
12565
12566
12567
12568
12569
12570
12571
12572
12573
12574
12575
12576
12577
12578
12579
12580
12581
12582
12583
12584
12585
12586
12587
12588
12589
12590
12591
12592
12593
12594
12595
12596
12597
12598
12599
12600
12601
12602
12603
12604
12605
12606
12607
12608
12609
12610
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620
12621
12622
12623
12624
12625
12626
12627
12628
12629
12630
12631
12632
12633
12634
12635
12636
12637
12638
12639
12640
12641
12642
12643
12644
12645
12646
12647
12648
12649
12650
12651
12652
12653
12654
12655
12656
12657
12658
12659
12660
12661
12662
12663
12664
12665
12666
12667
12668
12669
12670
12671
12672
12673
12674
12675
12676
12677
12678
12679
12680
12681
12682
12683
12684
12685
12686
12687
12688
12689
12690
12691
12692
12693
12694
12695
12696
12697
12698
12699
12700
12701
12702
12703
12704
12705
12706
12707
12708
12709
12710
12711
12712
12713
12714
12715
12716
12717
12718
12719
12720
12721
12722
12723
12724
12725
12726
12727
12728
12729
12730
12731
12732
12733
12734
12735
12736
12737
12738
12739
12740
12741
12742
12743
12744
12745
12746
12747
12748
12749
12750
12751
12752
12753
12754
12755
12756
12757
12758
12759
12760
12761
12762
12763
12764
12765
12766
12767
12768
12769
12770
12771
12772
12773
12774
12775
12776
12777
12778
12779
12780
12781
12782
12783
12784
12785
12786
12787
12788
12789
12790
12791
12792
12793
12794
12795
12796
12797
12798
12799
12800
12801
12802
12803
12804
12805
12806
12807
12808
12809
12810
12811
12812
12813
12814
12815
12816
12817
12818
12819
12820
12821
12822
12823
12824
12825
12826
12827
12828
12829
12830
12831
12832
12833
12834
12835
12836
12837
12838
12839
12840
12841
12842
12843
12844
12845
12846
12847
12848
12849
12850
12851
12852
12853
12854
12855
12856
12857
12858
12859
12860
12861
12862
12863
12864
12865
12866
12867
12868
12869
12870
12871
12872
12873
12874
12875
12876
12877
12878
12879
12880
12881
12882
12883
12884
12885
12886
12887
12888
12889
12890
12891
12892
12893
12894
12895
12896
12897
12898
12899
12900
12901
12902
12903
12904
12905
12906
12907
12908
12909
12910
12911
12912
12913
12914
12915
12916
12917
12918
12919
12920
12921
12922
12923
12924
12925
12926
12927
12928
12929
12930
12931
12932
12933
12934
12935
12936
12937
12938
12939
12940
12941
12942
12943
12944
12945
12946
12947
12948
12949
12950
12951
12952
12953
12954
12955
12956
12957
12958
12959
12960
12961
12962
12963
12964
12965
12966
12967
12968
12969
12970
12971
12972
12973
12974
12975
12976
12977
12978
12979
12980
12981
12982
12983
12984
12985
12986
12987
12988
12989
12990
12991
12992
12993
12994
12995
12996
12997
12998
12999
13000
13001
13002
13003
13004
13005
13006
13007
13008
13009
13010
13011
13012
13013
13014
13015
13016
13017
13018
13019
13020
13021
13022
13023
13024
13025
13026
13027
13028
13029
13030
13031
13032
13033
13034
13035
13036
13037
13038
13039
13040
13041
13042
13043
13044
13045
13046
13047
13048
13049
13050
13051
13052
13053
13054
13055
13056
13057
13058
13059
13060
13061
13062
13063
13064
13065
13066
13067
13068
13069
13070
13071
13072
13073
13074
13075
13076
13077
13078
13079
13080
13081
13082
13083
13084
13085
13086
13087
13088
13089
13090
13091
13092
13093
13094
13095
13096
13097
13098
13099
13100
13101
13102
13103
13104
13105
13106
13107
13108
13109
13110
13111
13112
13113
13114
13115
13116
13117
13118
13119
13120
13121
13122
13123
13124
13125
13126
13127
13128
13129
13130
13131
13132
13133
13134
13135
13136
13137
13138
13139
13140
13141
13142
13143
13144
13145
13146
13147
13148
13149
13150
13151
13152
13153
13154
13155
13156
13157
13158
13159
13160
13161
13162
13163
13164
13165
13166
13167
13168
13169
13170
13171
13172
13173
13174
13175
13176
13177
13178
13179
13180
13181
13182
13183
13184
13185
13186
13187
13188
13189
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
13212
13213
13214
13215
13216
13217
13218
13219
13220
13221
13222
13223
13224
13225
13226
13227
13228
13229
13230
13231
13232
13233
13234
13235
13236
13237
13238
13239
13240
13241
13242
13243
13244
13245
13246
13247
13248
13249
13250
13251
13252
13253
13254
13255
13256
13257
13258
13259
13260
13261
13262
13263
13264
13265
13266
13267
13268
13269
13270
13271
13272
13273
13274
13275
13276
13277
13278
13279
13280
13281
13282
13283
13284
13285
13286
13287
13288
13289
13290
13291
13292
13293
13294
13295
13296
13297
13298
13299
13300
13301
13302
13303
13304
13305
13306
13307
13308
13309
13310
13311
13312
13313
13314
13315
13316
13317
13318
13319
13320
13321
13322
13323
13324
13325
13326
13327
13328
13329
13330
13331
13332
13333
13334
13335
13336
13337
13338
13339
13340
13341
13342
13343
13344
13345
13346
13347
13348
13349
13350
13351
13352
13353
13354
13355
13356
13357
13358
13359
13360
13361
13362
13363
13364
13365
13366
13367
13368
13369
13370
13371
13372
13373
13374
13375
13376
13377
13378
13379
13380
13381
13382
13383
13384
13385
13386
13387
13388
13389
13390
13391
13392
13393
13394
13395
13396
13397
13398
13399
13400
13401
13402
13403
13404
13405
13406
13407
13408
13409
13410
13411
13412
13413
13414
13415
13416
13417
13418
13419
13420
13421
13422
13423
13424
13425
13426
13427
13428
13429
13430
13431
13432
13433
13434
13435
13436
13437
13438
13439
13440
13441
13442
13443
13444
13445
13446
13447
13448
13449
13450
13451
13452
13453
13454
13455
13456
13457
13458
13459
13460
13461
13462
13463
13464
13465
13466
13467
13468
13469
13470
13471
13472
13473
13474
13475
13476
13477
13478
13479
13480
13481
13482
13483
13484
13485
13486
13487
13488
13489
13490
13491
13492
13493
13494
13495
13496
13497
13498
13499
13500
13501
13502
13503
13504
13505
13506
13507
13508
13509
13510
13511
13512
13513
13514
13515
13516
13517
13518
13519
13520
13521
13522
13523
13524
13525
13526
13527
13528
13529
13530
13531
13532
13533
13534
13535
13536
13537
13538
13539
13540
13541
13542
13543
13544
13545
13546
13547
13548
13549
13550
13551
13552
13553
13554
13555
13556
13557
13558
13559
13560
13561
13562
13563
13564
13565
13566
13567
13568
13569
13570
13571
13572
13573
13574
13575
13576
13577
13578
13579
13580
13581
13582
13583
13584
13585
13586
13587
13588
13589
13590
13591
13592
13593
13594
13595
13596
13597
13598
13599
13600
13601
13602
13603
13604
13605
13606
13607
13608
13609
13610
13611
13612
13613
13614
13615
13616
13617
13618
13619
13620
13621
13622
13623
13624
13625
13626
13627
13628
13629
13630
13631
13632
13633
13634
13635
13636
13637
13638
13639
13640
13641
13642
13643
13644
13645
13646
13647
13648
13649
13650
13651
13652
13653
13654
13655
13656
13657
13658
13659
13660
13661
13662
13663
13664
13665
13666
13667
13668
13669
13670
13671
13672
13673
13674
13675
13676
13677
13678
13679
13680
13681
13682
13683
13684
13685
13686
13687
13688
13689
13690
13691
13692
13693
13694
13695
13696
13697
13698
13699
13700
13701
13702
13703
13704
13705
13706
13707
13708
13709
13710
13711
13712
13713
13714
13715
13716
13717
13718
13719
13720
13721
13722
13723
13724
13725
13726
13727
13728
13729
13730
13731
13732
13733
13734
13735
13736
13737
13738
13739
13740
13741
13742
13743
13744
13745
13746
13747
13748
13749
13750
13751
13752
13753
13754
13755
13756
13757
13758
13759
13760
13761
13762
13763
13764
13765
13766
13767
13768
13769
13770
13771
13772
13773
13774
13775
13776
13777
13778
13779
13780
13781
13782
13783
13784
13785
13786
13787
13788
13789
13790
13791
13792
13793
13794
13795
13796
13797
13798
13799
13800
13801
13802
13803
13804
13805
13806
13807
13808
13809
13810
13811
13812
13813
13814
13815
13816
13817
13818
13819
13820
13821
13822
13823
13824
13825
13826
13827
13828
13829
13830
13831
13832
13833
13834
13835
13836
13837
13838
13839
13840
13841
13842
13843
13844
13845
13846
13847
13848
13849
13850
13851
13852
13853
13854
13855
13856
13857
13858
13859
13860
13861
13862
13863
13864
13865
13866
13867
13868
13869
13870
13871
13872
13873
13874
13875
13876
13877
13878
13879
13880
13881
13882
13883
13884
13885
13886
13887
13888
13889
13890
13891
13892
13893
13894
13895
13896
13897
13898
13899
13900
13901
13902
13903
13904
13905
13906
13907
13908
13909
13910
13911
13912
13913
13914
13915
13916
13917
13918
13919
13920
13921
13922
13923
13924
13925
13926
13927
13928
13929
13930
13931
13932
13933
13934
13935
13936
13937
13938
13939
13940
13941
13942
13943
13944
13945
13946
13947
13948
13949
13950
13951
13952
13953
13954
13955
13956
13957
13958
13959
13960
1
```

Corrigé exercice : question 2

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 client.py
->
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
1 [| 0.7%] 5 [| 0.0%]
2 [| 0.0%] 6 [| 0.7%]
3 [| 0.7%] 7 [| 0.7%]
4 [| 1.3%] 8 [| 0.0%]
Mem[|||||] 1.586/7.666 Tasks: 116, 319 thr; 1 running
Swp[ 0K/9.31G] Load average: 0.51 0.36 0.19
Uptime: 03:04:12
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.

Corrigé exercice : question 2

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
1 [ | 0.7%] 5 [ 0.0%]
2 [ 0.0%] 6 [ | 0.7%]
3 [ | 0.7%] 7 [ | 0.7%]
4 [ 1.3%] 8 [ 0.0%]
Mem[||||||| 1.586/7.666] Tasks: 116, 319 thr; 1 running
Swp[ 0K/9.316] Load average: 0.51 0.36 0.19
Uptime: 03:04:12
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 client.py
-> 
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort By F7 Nice F8 Nicer F9 Kill F10 Quit

- 2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.
- 2.2 Quel(s) pourrait(aient) être le pid du parent du processus à l'exécution de client.py ?

Corrigé exercice : question 2

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
1 [ | 0.7%] 5 [ 0.0%]
2 [ 0.0%] 6 [ | 0.7%]
3 [ | 0.7%] 7 [ | 0.7%]
4 [ 1.3%] 8 [ 0.0%]
Mem[||||||| 1.586/7.666] Tasks: 116, 319 thr; 1 running
Swp[ 0K/9.316] Load average: 0.51 0.36 0.19
Uptime: 03:04:12
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:fiche_Les signaux$ python3 client.py
-> 
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort By F7 Nice F8 Nice+ F9 Kill F10 Quit

- 2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.
- 2.2 Quel(s) pourrait(aient) être le pid du parent du processus à l'exécution de client.py ?

Corrigé exercice : question 2

Terminal

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
pascal@aslin:~$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)
```

Terminal

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

1  [ |                               0.7%]  5  [                               0.0%]
2  [ |                               0.0%]  6  [ |                               0.7%]
3  [ |                               0.7%]  7  [ |                               0.7%]
4  [ |                               1.3%]  8  [                               0.0%]
Mem[|||||] 1.586/7.666 Tasks: 116, 319 thr; 1 running
Swp[      ] 0K/9.316   Load average: 0.51 0.36 0.19
                                Uptime: 03:04:12
```

Terminal

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
pascal@aslin:~$ python3 client.py
-> [ ]
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUN	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

- 2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.
- 2.2 Quel(s) pourrait(aient) être le pid du parent du processus à l'exécution de client.py ? Le parent est un programme **bash**, donc en utilisant les informations de **htop** on voit qu'il peut s'agir du pid 11806 ou 11792
- 2.3 Comment mettre fin au processus associé serveur.py ? (donnez deux possibilités)

Corrigé exercice : question 2

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:~$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
1 [ | 0.7%] 5 [ 0.0%]
2 [ 0.0%] 6 [ | 0.7%]
3 [ | 0.7%] 7 [ | 0.7%]
4 [ | 1.3%] 8 [ 0.0%]
Mem[|||||] 1.586/7.666 Tasks: 116, 319 thr; 1 running
Swp[ 0K/9.316] Load average: 0.51 0.36 0.19
Uptime: 03:04:12
```

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
pascal@aslin:~$ python3 client.py
-> [ ]
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUN	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort By F7 Nice F8 Nice+ F9 Kill F10 Quit

- 2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.
- 2.2 Quel(s) pourrait(aient) être le pid du parent du processus à l'exécution de client.py ? Le parent est un programme **bash**, donc en utilisant les informations de **htop** on voit qu'il peut s'agir du pid 11806 ou 11792
- 2.3 Comment mettre fin au processus associé serveur.py ? (donnez deux possibilités)

Corrigé exercice : question 2

```

Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
pascal@aslin:~$ python3 serveur.py
pid du serveur 11812
pid du père du processus 11800
Connexion de: ('127.0.0.1', 60876)

```

```

Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
1 [ | 0.7%] 5 [ 0.0%]
2 [ 0.0%] 6 [ | 0.7%]
3 [ | 0.7%] 7 [ | 0.7%]
4 [ | 1.3%] 8 [ 0.0%]
Mem[||||| 1.586/7.666] Tasks: 116, 319 thr; 1 running
Swp[ 0K/9.316] Load average: 0.51 0.36 0.19
Uptime: 03:04:12

```

```

Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
pascal@aslin:~$ python3 client.py
->

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUN	MEM%	TIME+	Command
11836	pascal	20	0	1043M	56308	44812	S	0.0	0.7	0:00.33	/usr/bin/gnome-cal
11819	pascal	20	0	8280	4088	3344	R	2.0	0.1	0:01.07	htop
11817	pascal	20	0	17832	9644	5812	S	0.0	0.1	0:00.03	python3 client.py
11812	pascal	20	0	17832	9780	5844	S	0.0	0.1	0:00.02	python3 serveur.py
11806	pascal	20	0	8112	4880	3344	S	0.0	0.1	0:00.01	bash
11800	pascal	20	0	8112	4864	3412	S	0.0	0.1	0:00.01	bash
11792	pascal	20	0	8244	5188	3520	S	0.0	0.1	0:00.03	bash
10235	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.00	/usr/libexec/xdg-de
10232	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10231	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.01	/usr/libexec/xdg-de
10229	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.00	/usr/libexec/xdg-de
10228	pascal	20	0	490M	23512	18044	S	0.0	0.3	0:00.12	/usr/libexec/xdg-de
10226	pascal	20	0	384M	13216	11448	S	0.0	0.2	0:00.04	/usr/libexec/xdg-de

- 2.1 Quel est le numéro pid du processus lié à l'exécution de client.py ? La commande **htop** nous indique que le pid du client.py est 11817.
- 2.2 Quel(s) pourrait(aient) être le pid du parent du processus à l'exécution de client.py ? Le parent est un programme **bash**, donc en utilisant les informations de **htop** on voit qu'il peut s'agir du pid 11806 ou 11792
- 2.3 Comment mettre fin au processus associé serveur.py ? (donnez deux possibilités) En tapant **Ctrl-C**, avec la fenêtre du terminal correspondant actif, en cliquant sur la croix de la fenêtre appropriée ou en tapant dans une console **kill -2 11812**.

Corrigé exercice : question 3

3. Tuez le serveur et continuez à envoyer des messages par le biais du client. Quelle message d'erreur apparaît ? En expliquer la raison. Aurait-on pu procéder différemment ?

Corrigé exercice : question 3

3. Tuez le serveur et continuez à envoyer des messages par le biais du client. Quelle message d'erreur apparaît ? En expliquer la raison. Aurait-on pu procéder différemment ?
- ▶ On a l'erreur suivante : **BrokenPipeError : [Errno 32] Broken pipe.**
En fermant le serveur, le canal de communication n'est plus disponible rendant impossible l'écriture des données dans le PIPE (canal, tuyau). Le programme client a reçu un signal **SIGPIPE**.

Corrigé exercice : question 3

3. Tuez le serveur et continuez à envoyer des messages par le biais du client. Quelle message d'erreur apparaît ? En expliquer la raison. Aurait-on pu procéder différemment ?
- ▶ On a l'erreur suivante : **BrokenPipeError : [Errno 32] Broken pipe.**
En fermant le serveur, le canal de communication n'est plus disponible rendant impossible l'écriture des données dans le PIPE (canal, tuyau). Le programme client a reçu un signal **SIGPIPE**.
 - ▶ Il n'est pas nécessaire d'arrêter le serveur pour casser le canal de communication, on peut également exécuter la commande : `kill -13 pid` (où pid est le numéro du processus à l'exécution du client).

Corrigé exercice : question 4

4. Modifier le programme client pour que le programme prenne en considération cette erreur en demandant à l'utilisateur s'il veut continuer ou s'il veut mettre fin au programme.

Corrigé exercice : question 4

```
# client.py
import socket, signal, sys, os

def gestionnaire(signum, stack):
    r = input("Un problème de communication avec le serveur.\nVoulez vous attendre (o/n)?")
    if r=='n':
        sys.exit(0)
    else:
        raise BrokenPipeError

print("pid du client", os.getpid())
host, port = socket.gethostname(), 5000
signal.signal(signal.SIGPIPE, gestionnaire)

client_socket = socket.socket()
client_socket.connect((host, port))

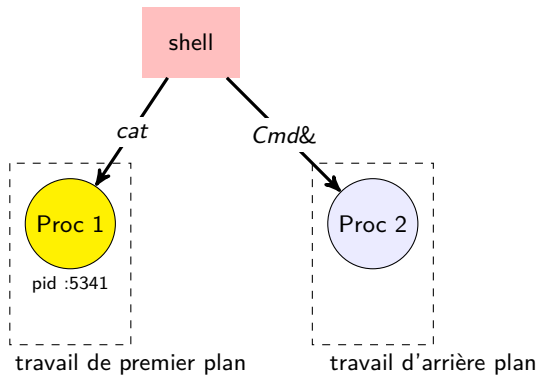
try:
    message = input("Taper votre message -> ") #message à envoyer

    while message.lower().strip() != 'fin':
        client_socket.send(message.encode()) # envoi un message
        data = client_socket.recv(1024).decode() # reception d'un message
        print('Reçu du serveur: ' + data) # affiche le message reçu
        message = input(" -> ") # nouveau message à envoyer
except BrokenPipeError:
    print("Le programme se poursuit")

finally:
    client_socket.close()
```

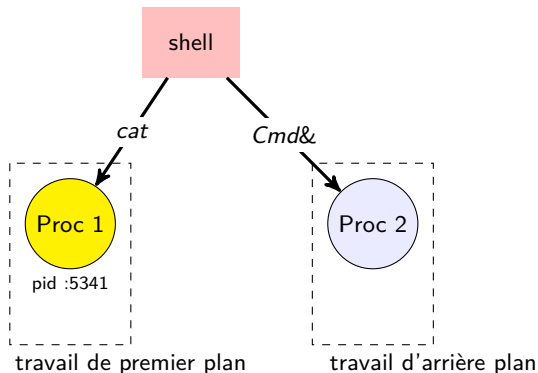
Description du fonctionnement CTRL-Z

Dans le terminal



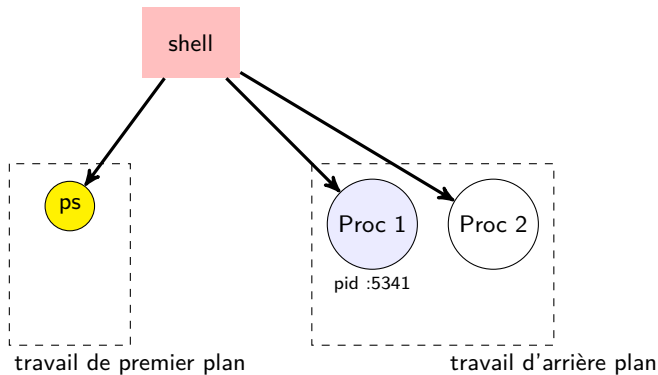
Description du fonctionnement CTRL-Z

Dans le terminal

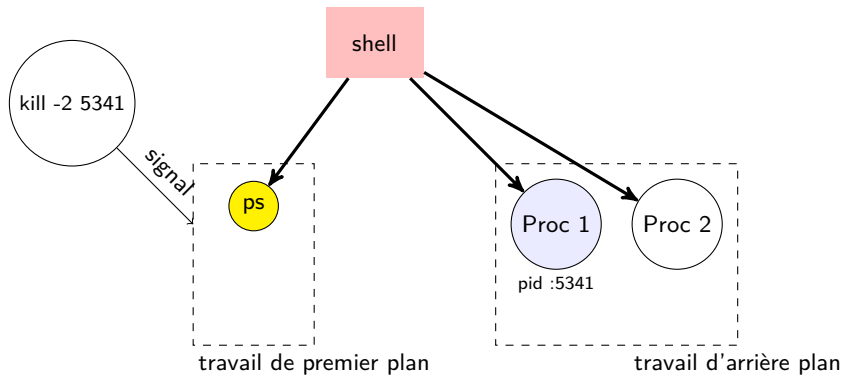


La commande **Ctrl-Z** va passer le processus **cat** dans les travaux en arrière plan et changer son état en **Stopped**. On aurait pu également taper la commande **kill SIGSTP 5341** (si 5341 est le pid du processus de **cat**). L'état **Stopped** du processus **cat** interdit au processus de s'exécuter. La reprise est assurée à la réception du signal **SIGCONT**.

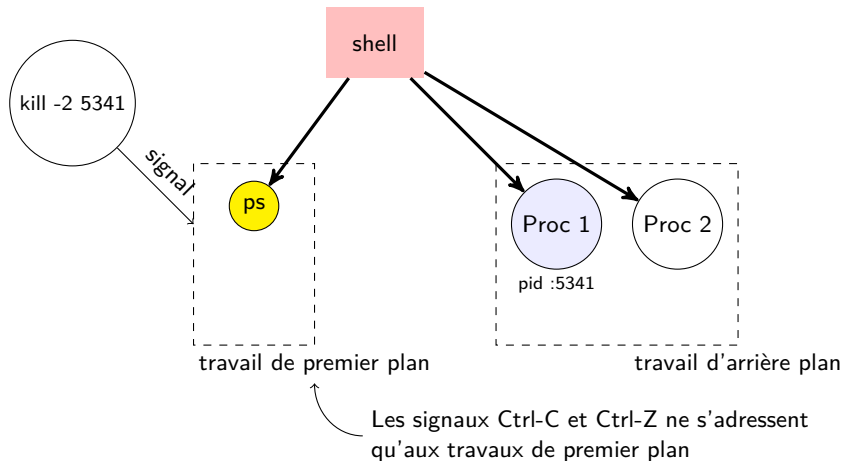
Description du fonctionnement CTRL-Z



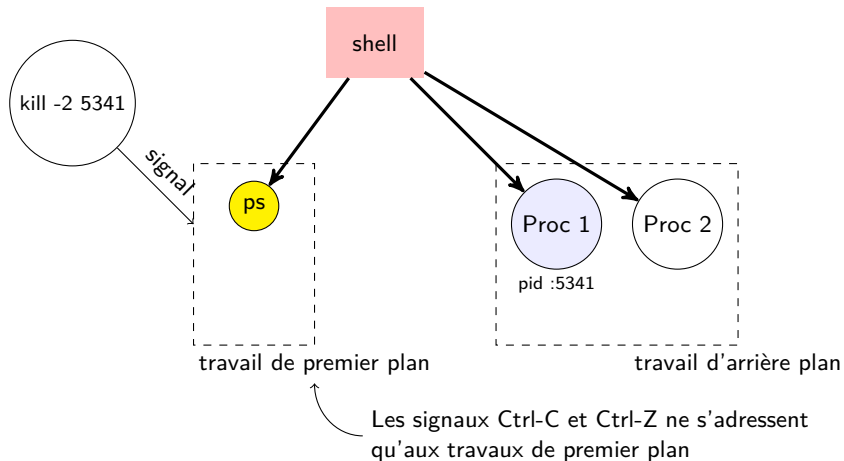
Description du fonctionnement CTRL-Z



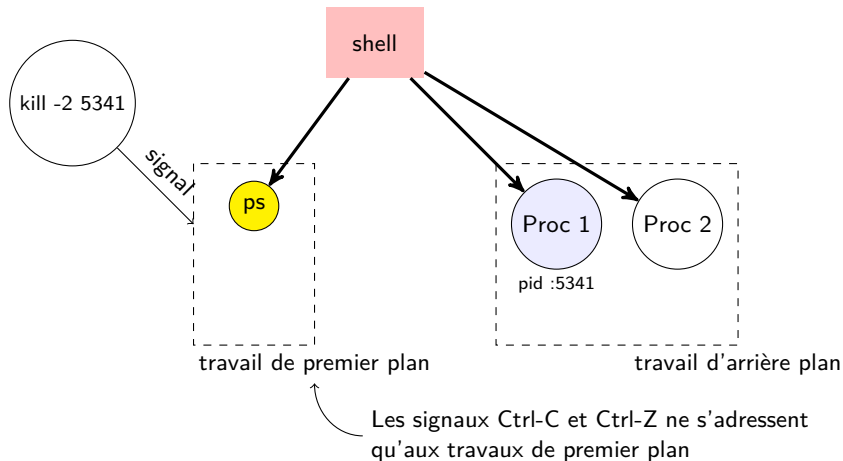
Description du fonctionnement CTRL-Z



Description du fonctionnement CTRL-Z



Description du fonctionnement CTRL-Z



La commande **killall -2 cat** n'a pas d'action sur le processus **cat** car il ne fait partie des processus qui sont au premier plan.

À l'aide de la commande **bg** (signal SIGCONT) le processus réintègre les processus de premier plan.

Description du fonctionnement CTRL-Z

