Tlemcen University
Department of Computer Science
1st Year Engineering
" Introduction to operating systems 1"

Sessions 10 : Process management

Academic year: 2023-2024

Outline

- Process concepts
- Process visualization
- Types of process
- Process management
- Process priority
- Operating modes
- Process management commands

Process concepts

Process

- *When a program is executed, the system creates a process by placing the program data and code in memory and creating an execution stack.*
- *A process is therefore an instance of a program with an associated processor environment (ordinal counter, registers, etc.) and memory environment.*
- *Each process has :*
  - *a **PID**: **P**rocess **ID**entifier, a unique process identifier ;*
  - *a **PPID**: **P**arent **P**rocess **ID**entifier, a unique identifier for the parent process.*

- *By successive filiations, the **init** process is the parent of all processes.*
  - *A process is always created by a parent process;*
  - *A parent process can have several child processes.*
- *There is a parent/child relationship between processes,*
- *A child process is the result of a system call to the fork() primitive by the parent process, which duplicates its own code to create a child.*
- *The child's PID is sent back to the parent process so that it can talk to it.*
- *Each child has its parent's identifier, the PPID.*

- *The PID represents the process at the time of execution.*
- *At the end of the process, the number is available again for another process.*
- *Executing the same command or program several times will produce a different PID each time.*
- *Each process is represented in the OS by a PCB (process control block).*

PCB	
Pointer	Process state
Process ID	
Program counter	
Registers	
Memory limits	
List of open files	
.....	

- PCB: contains information about a specific process, for example:
  - *The state of the process.*
  - *Instruction counter: indicates the address of the next instruction to be executed by this process.*
  - *Pointer: It is a stack pointer that is required to be saved when the process is switched from one state to another to retain the current position of the process.*
  - *CPU scheduling information: information about the priority of the process.*
  - *Memory management information: values of the base and limit registers, page tables or segment tables.*
  - *I/O status information: list of I/O devices allocated to this process, a list of open files, etc.*

Process visualization

- The command ***ps*** displays the status of current processes.
  - *ps command syntax:*

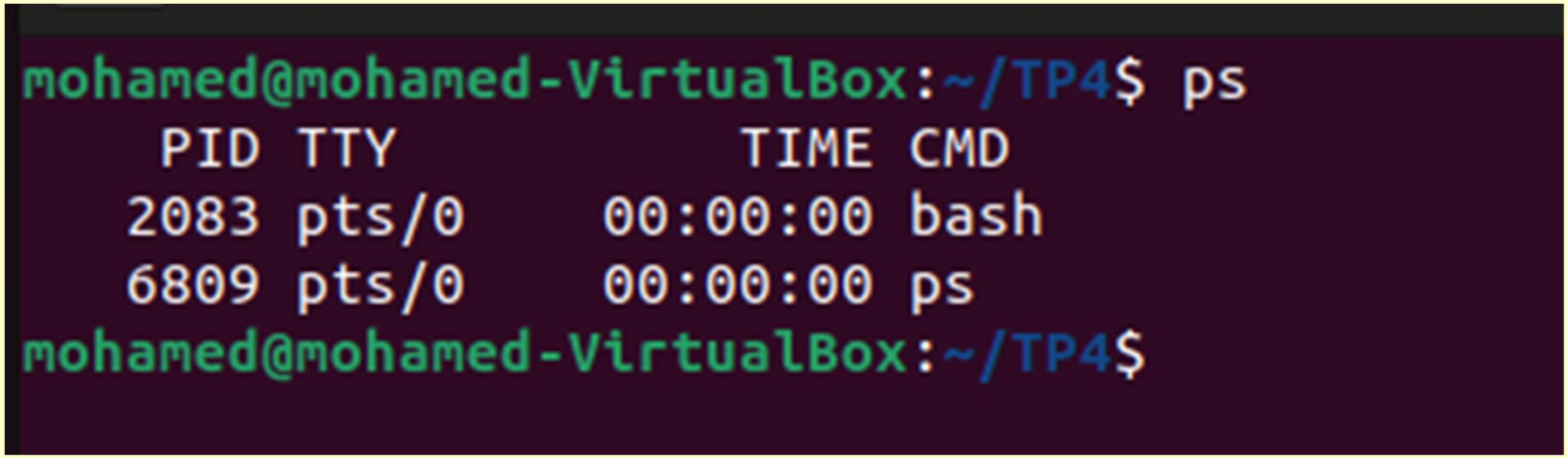
```
ps [-e] [-f] [-u login]
```

  - *Example:*

```
ps -fu root
```

  - *Main options for the ps command:*
    - *-e : Displays all processes.*
    - *-f: Displays additional information.*
    - *-u login: Displays the user's processes.*

- *Other options :*
  - *-g: Displays processes in the group.*
  - *-t tty: Displays processes running from the terminal.*
  - *-p PID: Displays process information.*
  - *-H: Displays information in the form of a tree structure.*
  - *-I: Displays additional information.*
  - *--sort COL: Sort the result according to a column.*
- *If no option is specified, the **ps** command will only display processes running from the current terminal.*


mohamed@mohamed-VirtualBox:~/TP4$ ps
PID TTY TIME CMD
2083 pts/0 00:00:00 bash
6809 pts/0 00:00:00 ps
mohamed@mohamed-VirtualBox:~/TP4$

```
mohamed@mohamed-VirtualBox:~/TP4$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 18:15 ?        00:00:01 /sbin/init splash
root           2       0  0 18:15 ?        00:00:00 [kthreadd]
root           3       2  0 18:15 ?        00:00:00 [rcu_gp]
root           4       2  0 18:15 ?        00:00:00 [rcu_par_gp]
root           5       2  0 18:15 ?        00:00:00 [slub_flushwq]
root           6       2  0 18:15 ?        00:00:00 [netns]
root           8       2  0 18:15 ?        00:00:00 [kworker/0:0H-events_highpri]
root          10       2  0 18:15 ?        00:00:00 [mm_percpu_wq]
root          11       2  0 18:15 ?        00:00:00 [rcu_tasks_kthread]
root          12       2  0 18:15 ?        00:00:00 [rcu_tasks_rude_kthread]
root          13       2  0 18:15 ?        00:00:00 [rcu_tasks_trace_kthread]
root          14       2  0 18:15 ?        00:00:00 [ksoftirqd/0]
root          15       2  0 18:15 ?        00:00:05 [rcu_preempt]
root          16       2  0 18:15 ?        00:00:00 [migration/0]
root          17       2  0 18:15 ?        00:00:00 [idle_inject/0]
root          19       2  0 18:15 ?        00:00:00 [cpuhp/0]
root          20       2  0 18:15 ?        00:00:00 [cpuhp/1]
root          21       2  0 18:15 ?        00:00:00 [idle_inject/1]
root          22       2  0 18:15 ?        00:00:00 [migration/1]
root          23       2  0 18:15 ?        00:00:00 [ksoftirqd/1]
```

- *UID: User owner.*
- *PID: Process IDentifier.*
- *PPID: Parent Process IDentifier.*
- *C: Process priority.*
- *STIME: Execution date and time.*
- *TTY: Terminal of execution.*
- *TIME: Processing time.*
- *CMD: Command executed.*

- *The **ps** command display can be customized.*
- ***Example***

```
ps -e --format "%P%p%c%n" --sort ppid --headers
```

```
mohamed@mohamed-VirtualBox:~/TP4$ ps -e --format "%P%p%c%n" --sort ppid --headers
   PPID     PID COMMAND          NI
      0       1 systemd          0
      0       2 kthreadd         0
      1     195 systemd-journal  -1
      1     253 systemd-udevd    0
      1     395 systemd-oomd     0
      1     404 systemd-resolve  0
      1     407 systemd-timesyn  0
      1     598 accounts-daemon  0
      1     599 acpid            0
      1     602 avahi-daemon     0
      1     603 cron             0
      1     605 dbus-daemon      0
      1     607 NetworkManager   0
```

Types of process

- The user process :
  - *is started from a terminal associated with an user,*
  - *accesses resources via requests or daemons.*
- The system process (daemon) :
  - *is started by the system,*
  - *is not associated with any terminal and its owner is a system user (often root),*
  - *is loaded at start-up, resides in memory and is waiting for a call,*
  - *is generally identified by the letter d associated with the process name.*

Process management

- *A process cannot be run indefinitely, as this would be to the detriment of other running processes and would prevent multitasking.*
- *The total processing time available is therefore divided into time slots and each process with a priority accesses the processor in sequence.*
- *The process will take on several states during its lifetime:*
  - ***ready**: waiting for the processor to be available,*
  - ***running**: accesses the processor,*
  - ***suspended**: waiting for an I/O (input/output),*
  - ***stopped**: waiting for a signal from another process,*
  - ***zombie**: If a terminated process cannot be unloaded from memory, for example if one of its children is not terminated, it goes into a state called zombie.*
  - ***dead**: the parent of the process kills its child.*

R	Running
I	Sleepy (> 20 s)
S	Sleepy (< 20 s)
D	Waiting for a disk operation
T	Interrupted
Z	Zombie

- The end-of-process sequence is as follows:
  - *Open files are closed,*
  - *Release of used memory,*
  - *Sending a signal to the parent and child processes.*
- When a parent process dies, its children are orphaned.
  - *These are then adopted by the **init** process, which is responsible for destroying them.*

Process priority

- The processor works on a time-sharing basis, with each process occupying a quantum of processor time.
- Processes are classified by priority, the value of which varies from:
  - *-20 (the highest priority) to +20 (the lowest priority).*
  - *The default priority of a process is 0.*

Operating modes

- Processes can operate in two ways:
  - *Synchronous :*
    - *The user cannot access the shell while the command is being executed.*
    - *The command prompt reappears at the end of the process.*
  - *Asynchronous :*
    - *The process is run in the background,*
    - *The command prompt is re-displayed immediately*

Process management commands

The command *kill*

- *The command kill sends a stop signal to a process.*
- *Syntax of the command kill:*

```
kill [-signal] PID
```

- *Example:*

```
kill -9 1509
```

- *Codes for the main process stop signals*

Code	Signal	Description
2	*SIGINT*	Process interruption (CTRL+D)
9	*SIGKILL*	Immediate end of process
15	*SIGTERM*	End of process
18	*SIGCONT*	Restarting the process
19	*SIGSTOP*	Suspending the process

- Signals are the means of communication between processes.
- The kill command is used to send a signal to a process.
- The full list of kill signals is available by typing the command :

```
man 7 signal
```

- [CTRL] + [Z]
  - *By typing [CTRL+Z], the synchronous process is temporarily suspended.*
  - *Access to the prompt is restored after displaying the number of the process that has just been suspended.*
- Command &
  - *The & instruction executes the command in asynchronous mode (the command is then called a job) and displays the job number.*
  - *Access to the prompt is then restored.*

- ***Example:***

```
time ls -lR / > list.ls 2> /dev/null &
```

- *The job number is obtained during background processing and is displayed in square brackets, followed by the PID number.*

```
mohamed@mohamed-VirtualBox:~/TP4$ time ls -lR / > list.ls 2>/dev/null &
[1] 9691
mohamed@mohamed-VirtualBox:~/TP4$
real    0m7,482s
user    0m2,620s
sys     0m4,571s
```

Character string conversion: tr ((TRanslate)

The commands fg and bg

- *The command fg brings the process to the foreground:*
- ***Example**:*

```
$time ls -lR / > list.ls 2>/dev/null &
$fg 1
$time ls -lR / > list.ls 2/dev/null
```

- *The command bg places it in the background:[CTRL]+[Z]*
- *A process can be placed in the background when it is created using the & argument or later using the [CTRL+Z] keys,*
- *It can be brought back to the foreground using the fg command and its job number.*

```
mohamed@mohamed-VirtualBox:~/TP4$ time ls -lR / > list.ls 2>/dev/null &
[1] 10025
mohamed@mohamed-VirtualBox:~/TP4$ fg 1
time ls --color=auto -lR / > list.ls 2> /dev/null

real    0m7,145s
user    0m2,407s
sys     0m4,555s
mohamed@mohamed-VirtualBox:~/TP4$
```

```
mohamed@mohamed-VirtualBox:~/TP4$ time ls -lR / > list.ls 2>/dev/null
^Z
[1]+  Arrêté                 ls --color=auto -lR / > list.ls 2> /dev/null

real    0m1,929s
user    0m0,000s
sys     0m0,000s
mohamed@mohamed-VirtualBox:~/TP4$ bg 1
[1]+ ls --color=auto -lR / > list.ls 2> /dev/null &
mohamed@mohamed-VirtualBox:~/TP4$
```

The command *jobs*

- *The command jobs displays the list of processes running in the background and specifies their job number.*
- *Example:*

```
mohamed@mohamed-VirtualBox:~/TP4$ time ls -lR / > list.ls 2>/dev/null
^Z
[1]+  Arrêté                 ls --color=auto -lR / > list.ls 2> /dev/null

real    0m1,025s
user    0m0,000s
sys     0m0,000s
mohamed@mohamed-VirtualBox:~/TP4$ time ls -lR / > list.ls 2>/dev/null &
[2] 11298
mohamed@mohamed-VirtualBox:~/TP4$ jobs
[1]+  Arrêté                 ls --color=auto -lR / > list.ls 2> /dev/null
[2]-  En cours d'exécution   time ls --color=auto -lR / > list.ls 2> /dev/null &
mohamed@mohamed-VirtualBox:~/TP4$
real    0m7,734s
user    0m2,794s
sys     0m4,635s
```

- *The columns represent :*
  - *the job number ;*
  - *process order*
    - *+ : the next process to run by default with fg or bg,*
    - *- : the next process to take the +,*
  - *Running or Stopped (process suspended),*
  - *the command*

The commands *nice* and *renice*

- *The command **nice** is used to execute a command, specifying its priority.*
- *Syntax of the nice command*

```
$nice command priority
```

- *Example:*

```
$ nice -n +10 find / -name "file
```

- *Unlike root, a standard user can only reduce the priority of a process.*
  - *Only values between +0 and +19 will be accepted.*
  - *This constraint can be lifted by user or by group by modifying the file "/etc/security/limits.conf".*

- *The command renice is used to change the priority of a running process.*
- *Syntax of the renice command*

```
$renice priority [-g GID] [-p PID] [-u UID]
```

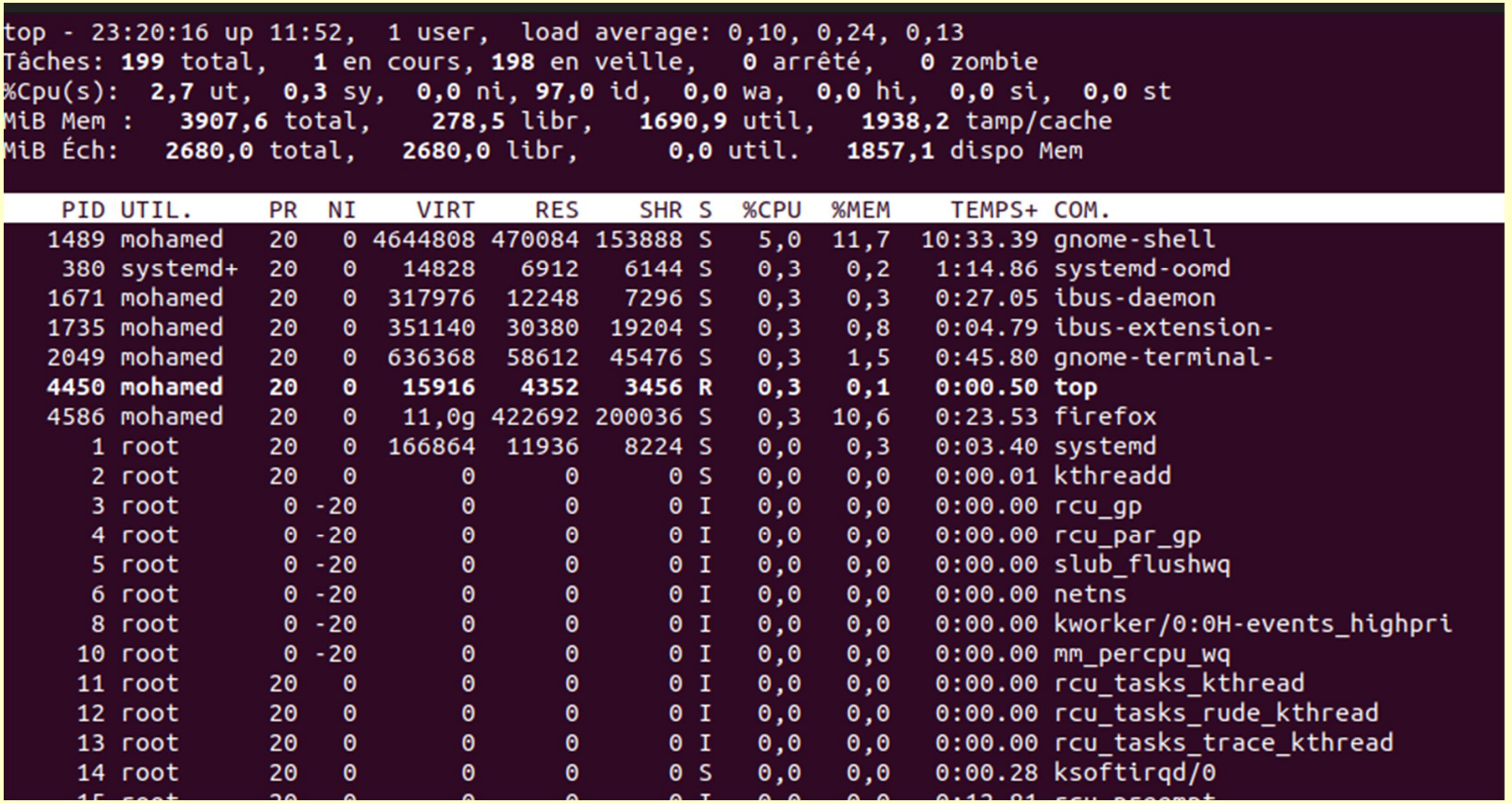
- *Main options for the command renice*
  - *-g : GID of the process owner group.*
  - *-p : PID of the process.*
  - *-u :UID of the process owner.*
- *Example:*

```
$ renice +15 -p 1652
```

- *The renice command acts on processes already running.*
- *It is therefore possible to change the priority of a specific process or several processes belonging to a user or group.*

The command *top*

- *The command top displays the processes and their resource consumption.*
- *The command top is used to control processes in real time and in interactive mode.*


top - 23:20:16 up 11:52, 1 user, load average: 0,10, 0,24, 0,13
Tâches: 199 total, 1 en cours, 198 en veille, 0 arrêté, 0 zombie
%Cpu(s): 2,7 ut, 0,3 sy, 0,0 ni, 97,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 3907,6 total, 278,5 libr, 1690,9 util, 1938,2 tamp/cache
MiB Éch: 2680,0 total, 2680,0 libr, 0,0 util. 1857,1 dispo Mem
PID UTIL. PR NI VIRT RES SHR S %CPU %MEM TEMPS+ COM.
1489 mohamed 20 0 4644808 470084 153888 S 5,0 11,7 10:33.39 gnome-shell
380 systemd+ 20 0 14828 6912 6144 S 0,3 0,2 1:14.86 systemd-oomd
1671 mohamed 20 0 317976 12248 7296 S 0,3 0,3 0:27.05 ibus-daemon
1735 mohamed 20 0 351140 30380 19204 S 0,3 0,8 0:04.79 ibus-extension-
2049 mohamed 20 0 636368 58612 45476 S 0,3 1,5 0:45.80 gnome-terminal-
4450 mohamed 20 0 15916 4352 3456 R 0,3 0,1 0:00.50 top
4586 mohamed 20 0 11,0g 422692 200036 S 0,3 10,6 0:23.53 firefox
1 root 20 0 166864 11936 8224 S 0,0 0,3 0:03.40 systemd
2 root 20 0 0 0 0 S 0,0 0,0 0:00.01 kthreadd
3 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_par_gp
5 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 slub_flushwq
6 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 netns
8 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/0:0H-events_highpri
10 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 I 0,0 0,0 0:00.00 rcu_tasks_kthread
12 root 20 0 0 0 0 I 0,0 0,0 0:00.00 rcu_tasks_rude_kthread
13 root 20 0 0 0 0 I 0,0 0,0 0:00.00 rcu_tasks_trace_kthread
14 root 20 0 0 0 0 S 0,0 0,0 0:00.28 ksoftirqd/0

Information returned by the command top

Column	Description
PID	Process identifier
USER	User owner
PR	Process priority
NI	Value of nice
%CPU	CPU load
%MEM	Memory load
TIME+	CPU usage time
COMMAND	The command executed

The commands pgrep and pkill

- *The command pgrep searches the running processes for a process name and displays the PIDs corresponding to the selection criteria on the standard output.*
- *The command pkill sends the indicated signal (by default SIGTERM) to each process.*
- *Syntax of the commands pgrep and pkill*

```
pgrep process
pkill [-signal] process
```

- *Examples:*
  - *To retrieve the sshd process number*

```
pgrep -u root sshd
```

  - *To kill all tomcat processes :*

```
pkill tomcat
```