Taskset Command Cheatsheet

A quick reference guide for the Linux `taskset` command, used to set or retrieve the CPU affinity of a process.

Taskset Basics and Usage

What is taskset?

`taskset` is a command-line utility in Linux that allows you to retrieve or set the CPU affinity of a process. CPU affinity dictates which CPU or set of CPUs a process is allowed to run on.
Setting CPU affinity can be useful for: • Performance tuning (e.g., restricting a CPU-intensive task to specific cores). • Troubleshooting (e.g., isolating a process to avoid interference). • Ensuring real-time tasks run on dedicated cores.
Processes inherit the CPU affinity mask from their parent process by default.
The affinity mask is a bitmask representing the set of CPUs on which the process may run. Bit 0 corresponds to CPU 0, bit 1 to CPU 1, and so on.
CPUs are numbered starting from 0.
The affinity setting is only valid for the process and its children (unless explicitly changed by the children).
The `taskset` command modifies the scheduler affinity mask, which is managed by the `sched_setaffinity` system call.

Syntax Overview

Retrieve affinity of an existing PID: `taskset [options] -p PID`
Launch a new command with a specific affinity: `taskset [options] MASK COMMAND [ARG...]`
Set affinity for an existing PID: `taskset [options] MASK -p PID`
`MASK` is a bitmask, typically represented as a hexadecimal number (e.g., `0x00000001` for CPU 0).
`PID` is the process ID (a number).
`[options]` are optional flags like `-p` (operate on PID) or `-c` (use CPU list instead of mask).
`COMMAND [ARG...]` is the command and its arguments to be executed with the specified affinity.

Representing CPU Masks/Lists

Hexadecimal Mask	A bitmask where the N-th bit (starting from 0) is set if the process can run on CPU N. • `0x1`: CPU 0 (binary 0001) • `0x3`: CPU 0, 1 (binary 0011) • `0x4`: CPU 2 (binary 0100) • `0xf`: CPU 0, 1, 2, 3 (binary 1111)
CPU List (using `-c`)	A comma-separated list of CPU numbers or ranges. • `0`: CPU 0 • `0,1,2`: CPU 0, 1, and 2 • `0-3`: CPU 0 through 3 • `1,3,5-7`: CPU 1, 3, and 5 through 7
Choosing Representation	The CPU list (`-c`) is often more human-readable for specific core assignments, while the hexadecimal mask is the native representation and can be useful for large numbers of CPUs or scripting.

Retrieving CPU Affinity (`-p`)

Syntax	`taskset -p PID`

Example: Get affinity of PID 1	`taskset -p 1`
	`pid 1's current affinity mask: ffffffff`
	
	(This example output shows affinity for all CPUs up to 31 on a 32-bit system, assuming `ffffffff` is the mask for all available CPUs).

Using `-c` with `-p`	Use `-c` to see the affinity as a list of CPU cores instead of a mask.
	`taskset -p -c 1`
	`pid 1's current affinity list: 0-31`
	
	(Example for a system with 32 CPUs)

Finding a process PID	Use commands like `pgrep`, `ps aux | grep`, or `htop` to find the PID of a running process.
Example: Find PID and get affinity for 'myprocess'	`PID=$(pgrep myprocess)` `taskset -p $PID`

Setting Affinity and Examples

Setting Affinity for a New Command

Syntax (Hex Mask)	`taskset MASK COMMAND [ARG...]`
Syntax (CPU List)	`taskset -c LIST COMMAND [ARG...]`
Example: Run `mycommand` only on CPU 0 (hex mask)	`taskset 0x1 mycommand`
Example: Run `mycommand` only on CPU 0 (CPU list)	`taskset -c 0 mycommand`
Example: Run `mycommand` on CPU 0 and CPU 1 (hex mask)	`taskset 0x3 mycommand`
Example: Run `mycommand` on CPU 0 and CPU 1 (CPU list)	`taskset -c 0,1 mycommand`
Example: Run `mycommand` on CPU 0 through CPU 3 (CPU list)	`taskset -c 0-3 mycommand`
Verification	You can verify the affinity of the newly launched process using `taskset -p <new_pid>`.

Setting Affinity for a Running Process

Syntax (Hex Mask)	`taskset MASK -p PID`
Syntax (CPU List)	`taskset -c LIST -p PID`
Example: Restrict PID 1234 to CPU 0 (hex mask)	`taskset 0x1 -p 1234`
Example: Restrict PID 1234 to CPU 0 (CPU list)	`taskset -c 0 -p 1234`
Example: Allow PID 1234 to run on CPU 0, 2, and 4 (CPU list)	`taskset -c 0,2,4 -p 1234`
Permissions	You typically need sufficient privileges (e.g., root) to set the affinity of processes owned by other users. You can usually set the affinity for your own processes.

Tips and Best Practices

Be Cautious with Critical Processes: Avoid restricting essential system processes unless you know exactly what you're doing, as it could impact system stability.
Check System CPU Count: Use `nproc` or look at `/proc/cpuinfo` to understand the available CPU cores on your system.
Understand Hyper-threading: If your system uses hyper-threading, remember that logical CPUs are numbered sequentially (e.g., 0, 1, 2, 3 on a dual-core CPU with HT). Cores 0 and 1 might be logical threads on one physical core, while 2 and 3 are on another. Restricting a process to CPU 0 and 1 might keep it on a single physical core, which might not be the performance gain you expect.
Verify Affinity: Always use `taskset -p <PID>` (or `taskset -p -c <PID>`) after setting affinity to confirm it was applied correctly.
Combine with `nice` : You can use `taskset` in conjunction with `nice` or `ionice` to manage both CPU affinity and process priority/scheduling class.
Persistent Affinity: `taskset` sets affinity for a running process or a newly launched command. This setting is not persistent across reboots or process restarts. For persistent settings, consider using systemd unit files or other service management configurations.
Checking which CPU a process is currently running on: Use `ps -o pid,comm,psr -p <PID>`. The `psr` column shows the Processor ID (CPU core) the process is currently running on.

Common Options

`taskset -a`	Affect all threads of the given PID. By default, `taskset -p` only affects the main thread.
`taskset -c, --cpu-list`	Interpret MASK as a comma-separated list of CPU numbers and ranges, rather than a bitmask.
`taskset -p, --pid`	Operate on the specified PID instead of launching a new command.
`taskset -h, --help`	Display help information.
`taskset -V, --version`	Display version information.