

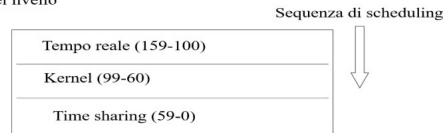
Scheduling 4 Real Time

martedì 28 ottobre 2025 12:24

Andiamo a vedere i processi real-time in ambito UNIX :

Caratteristiche

- 160 livelli di priorità
- 3 classi di priorità: Tempo Reale (159-100), Kernel (99-60), Time-Sharing (59-0)
- **kernel preemptable** (identificazione di safe places)
- **bitmap** per determinare i livelli non vuoti
- quanto di tempo variabile in funzione della classe e, in alcune classi, del livello



Attenzione al **kernel preemptable** : il kernel può perdere il controllo della CPU , cedendola ad altro thread . Quindi vediamo come usarla in UNIX, usando lo schema esteso :

```
#include <sched.h>

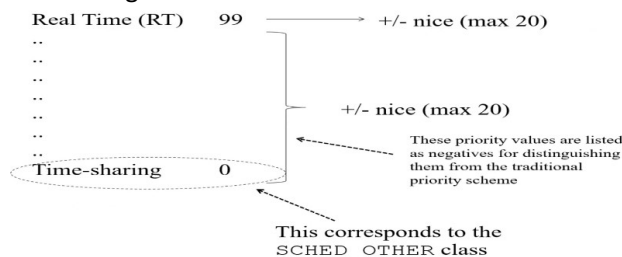
int sched_setscheduler(pid_t pid, int policy,
    const struct sched_param *p);

int sched_getscheduler(pid_t pid);

struct sched_param {
    ...
    int sched_priority;
    ...
};
```

From the shell: **chrt** command

Il quale schema esteso è il seguente :



Dove la classe 0-esima rappresenta quanto già detto appartiene ad una delle 40 (scheduler + gestione UNIX tradizionale). Se invece un thread non appartiene alla classe time-sharing, si ha che la nice rimane invariata (priorità di riferimento) la quale non viene impattata dal tempo di esecuzione . Un esempio è il comando TOP :

10879	dantele	20	0	6890648	350984	183144	S	72.7	2.2	72:56.01	vlc
793	root	-51	0	0	0	0	S	9.1	0.0	21:40.97	lrq/171-rtw88_pci
1225	avahi	20	0	11576	6480	3840	S	9.1	0.0	17:46.54	avahi-daemon
2471	dantele	9	-11	138456	34288	11488	S	9.1	0.2	2:01.26	pipewire-pulse
2784	dantele	20	0	4819272	353624	150188	S	9.1	2.2	15:31.33	gnome-shell
11834	root	0	-20	0	0	0	I	9.1	0.0	0:06.89	worker/u65:0-i915_flip
15128	dantele	20	0	12164	5824	3648	R	9.1	0.0	0:00.02	top
1	root	20	0	24172	14584	9128	S	0.0	0.1	0:04.15	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-rcu_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-sync_wq
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-kfree_rcu_reclaim
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-slab_flushwq
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R:0H-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:01.06	worker/R:0H-events_freezable
13	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/R-nm_percpu_wq
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
16	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
17	root	20	0	0	0	0	S	0.0	0.0	0:00.79	ksoftirqd/0
18	root	20	0	0	0	0	I	0.0	0.0	0:47.70	rcu_preempt
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_gp_kthread_worker/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.01	rcu_exp_gp_kthread_worker/0
21	root	rt	0	0	0	0	S	0.0	0.0	0:03.52	migration/0
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
25	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/2
26	root	rt	0	0	0	0	S	0.0	0.0	0:03.16	migration/2
27	root	20	0	0	0	0	S	0.0	0.0	0:00.06	ksoftirqd/2
28	root	20	0	0	0	0	I	0.0	0.0	0:00.84	worker/2:0H-events_long
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/2:0H-events_highpri
30	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/4
31	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/4
32	root	rt	0	0	0	0	S	0.0	0.0	0:02.24	migration/4
33	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftirqd/4
35	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	worker/4:0H-events_highpri
36	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/6
37	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/6
38	root	rt	0	0	0	0	S	0.0	0.0	0:02.32	migration/6
39	root	20	0	0	0	0	S	0.0	0.0	0:00.06	ksoftirqd/6

Dove i valori negativi si intende il livello assoluto +1 , mentre rt si significa che siamo nel livello real-time . Per cambiare il livello di priorità si usa il **chrt con privilegi d root -> chrt -p pid**. Nota : nello scheduling

UNIX si parla di **epoche di scheduling** ovvero un periodo di tempo per l'operatività del sistema . Ad ogni inizio di epoca ad ogni thread viene assegnato un numero di quanti di tempo da poter spendere. Quindi per ogni thread viene assegnato "una parte" del tempo di CPU del padre . Tutto ciò porta ad avere la possibilità di poter eseguire thread di classe di priorità bassa se pur con minimi quanti di tempo assegnati ad esse per ciascuna epoca (evitando la starvation).