Contingut

- Nice
- Gestió avançada de prioritats
  - Funcionament del Planificador Linux
  - Crides a Sistema relacionades
- Nohup

Nice 1/2

- **`nice`** executa una comanda (programa) modificant (augmentant o disminuint-li) la prioritat.
- El rang de prioritats va des de -20 (prioritat més alta) fins 19 (més baixa).
- Per defecte, es disminueix la prioritat en 10.
- Els usuaris normals només poden decrementar la prioritat. Augmentar la prioritat sol ho pot fer `root`.

Format de la comanda:

```
# nice [-n increment/decrement] [COMANDA [ARG]...]
```

on `increment` = sencer negatiu i `decrement` = sencer positiu.

Nice 2/2

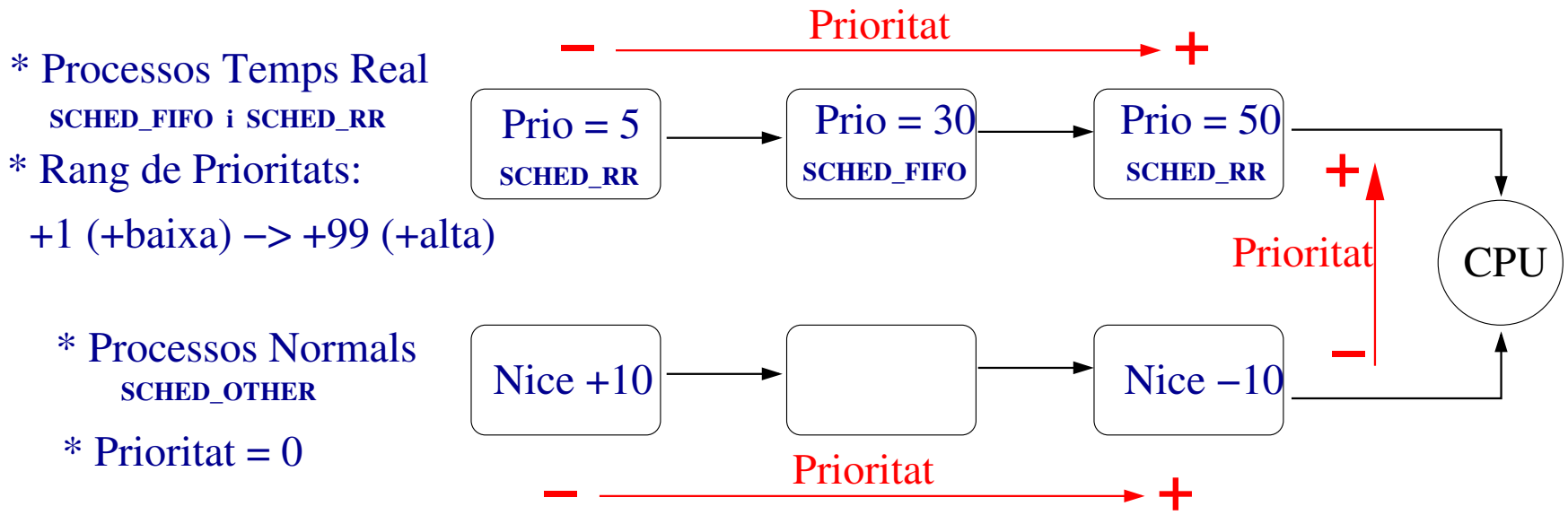
Suposem que disposem de dos scripts idèntics (`escriure1` i `escriure`):

```
#!/bin/bash
while : ; do let "a = ($a + 1) % 10" ; echo -n "$a" ; done
```

EXEMPLES:

```
$ escriure &
$ top       PID   PRI   NI   STAT   %CPU   TIME   COMMAND
            1002  25    0    R      70.2   1:27   escriure
$ nice -n -10 escriure &
nice:  no se puede establecer la prioridad:  Permiso denegado
$ nice -n 10 escriure &
$ top       PID   PRI   NI   STAT   %CPU   TIME   COMMAND
            1052  35    10   R      30.2   1:27   escriure
# nice escriure& nice -n -10 escriure1 &
$ top       PID   PRI   NI   STAT   %CPU   TIME   COMMAND
            3840  15    -10  R      46.5   1:14   escriure1
            3839  35    10   R      14.9   0:25   escriure
```

Gestió avançada de prioritats - Funcionament del Planificador Linux



Polítiques de planificació:

SCHED_FIFO: FIFO (molt perillós)

SCHED_RR: Round Robin

SCHED_OTHER: Round Robin però quan tots els processos han gastat 1 Quantum (210ms) es replanifica de nou tornant a donar a tots els processos 1 Quantum més d'execució. Assegura que cap procés entri en inanició.

Gestió avançada de prioritats - Crides a sistema relacionades

- Modificar la política i prioritat de planificació d'un procés:

```
int sched_setscheduler(pid_t pid, int politica, const struct \
sched_param *p);
```

- Obtenció de la política de planificació d'un procés:

```
int sched_getscheduler(pid_t pid); // retorna la política
de planificació
```

- Obtenció de la prioritat de planificació d'un procés:

```
int sched_getparam(pid_t pid, const struct_param *param);
// en param
```

- Obtenint la prioritat màxima i mínima d'una política de planificació:

```
int sched_get_priority_max(int politica);
int sched_get_priority_min(int politica);
```

- Abandonar la CPU:

```
int sched_yield(void);
```

Nohup

```
#!/bin/bash
# Nohup. Script ≡ comanda nohup. Exemple: $ nohup escriure &
# Continua l'execució de la comanda en background tot i abandonar el sistema.
# Sortida standard (i també d'errors) readreçades a nohup.out
trap " HUP          # ignora senyal hangup. No hi ha cap funció associada
exec 0< /dev/null   # desconnecta l'entrada standard
if [ -t 1 ]; then   # la sortida stàndard està associada a un terminal?
   if [ -w . ]; then    # l'usuari té permís d'escriptura en ./
         echo 'Enviant sortida a nohup.out'
         exec >> nohup.out   # readreça sortida standard
   else echo "Enviant sortida a $HOME/nohup.out" && exec >> $HOME/nohup.out
   fi
fi
# readreça sortida d'errors si es necessari
[ -t 2 ] && echo 'Enviant sortida derrors a nohup.out' && exec 2>&1
$@                  # s'executa la comanda
exit 0
```