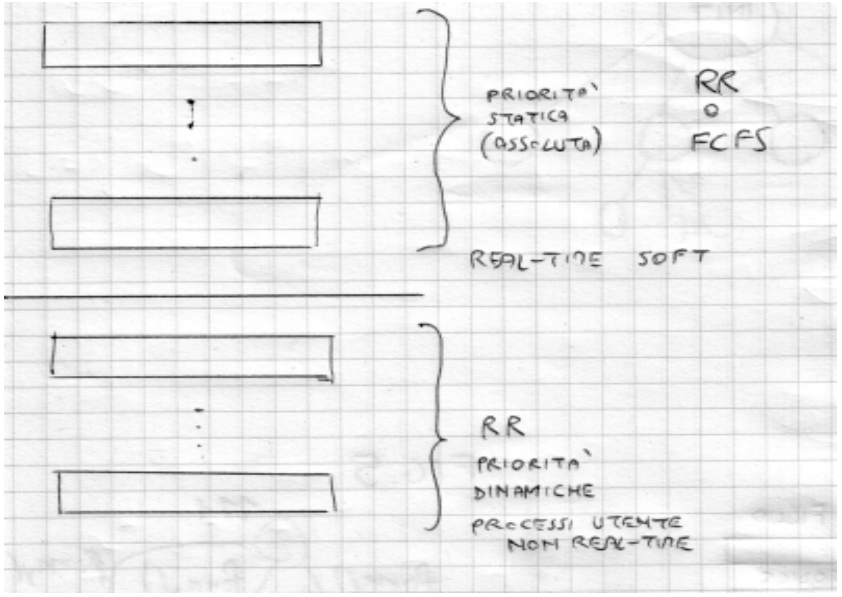
Appunti di sistemi operativi

Appunti per il corso universitario di sistemi operativi, riferito a sistemi Unix/Windows.
Si discute su problemi di sincronizzazione, memoria e scheduling dei processi.

SCHEDULING IN LINUX

in linux abbiamo 2 serie di code in base al tipo di processo:

- 

real-time: lo scheduler può essere RR o FCFS
- processi utente: non sono real-time e vengono gestiti da uno scheduler RR

crediti = **tick** = unità di tempo di CPU che il processo può utilizzare: ogni processo ha un insieme di crediti che può spendere

in particolare:

1. quando il processo diventa pronto o in attesa, si sottrae il numero di tick utilizzati
2. al momento del dispatch il processo pronto con più crediti viene eseguito
3. quando tutti i processi pronti hanno esaurito i crediti, questi vengono riassegnati:

$$\forall \text{ processo } i \; CREDITI\ i = \frac{CREDITI}{2} + PRIORITÀ$$

quando **tutti i processi eseguibili** hanno finito i crediti **vado a riassegnarli a tutti i processi (quindi anche a quelli in attesa)**: secondo questa equazione, i processi che erano in attesa acquisiranno un numero di crediti più alto

infatti per la **priorità** utilizziamo un numero che è proporzionale alla priorità (+ alto -> priorità + alta)

+ alta)

la bontà di un processo (**goodness**) è calcolata tramite i crediti, ai processi real-time viene aggiunta mille alla priorità

c'è starvation?

siccome la condizione prevede che tutti i processi pronti abbiano esaurito i crediti, non c'è starvation sui processi utente, a meno che non ci sia un flusso di nuovi processi ad alta priorità (real-time)

es. (per vedere che si tratta veramente di sistema a priorità assoluta)

```
#include <stdio.h>
#include <sched.h>

main() {
  struct sched_param p;
  int i;

  sched_getparam(0,&p);
  p.sched_priority = 99;
  sched_setscheduler(0, SCHED_RR, &p);
  sched_setparam(0, &p);
  for(i=0; i<1000000000; i++) {}
}
```

le istruzioni (1) e (2) possono essere eseguite solo da root (ma se non mettiamo un pezzo di codice che verifichi che l'operazione sia andata a buon fine non ce ne accorgiamo)

se lo eseguiamo da root, questo processo toglie il controllo del sistema all'utente; questo succede sia se la coda real-time è controllata da uno scheduler FCFS, sia da uno RR

continua..