Tutoraggio di Sistemi Operativi

Matteo Federico

Lezione #4

TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

In the last Episode

Gara di corsa

Scrivere un programma C multi-thread che simula una gara di corsa.
Il programma deve supportare un numero N di corridori, ognuno rappresentato da un thread separato.
Ogni thread/corridore, alla partenza, deve attendere un numero casuale di secondi, SOLO IL PRIMO a terminare è dichiarato vincitore della gara dal main thread.
Gli altri terminano senza fare niente.
Non sono richiesti meccanismi di sincronizzazione tra i thread, ignorare situazioni anomale causate da variabili condivise.

TOR VERGATA

Soluzione caso base

```c
int winner = 0;
void* runner(void* arg) {
    int id = *(int*)arg;
    int wait_time = rand() % 5 + 1;
    sleep(wait_time);
    if (winner == 0) {
        winner = id;
    }
    return NULL;
}
int main() {
    srand(0);
    pthread_t threads[N];
    int ids[N];
    for (int i = 0; i < N; i++) {
        ids[i] = i + 1;
        pthread_create(&threads[i], NULL, runner, &ids[i]);
    }
    for (int i = 0; i < N; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Corridore %d ha vinto la gara!\n", winner);
    return 0;
}

```

Dov'é il problema?

```
1  int winner = 0;
2  void* runner(void* arg) {
3      int id = *(int*)arg;
4      int wait_time = rand() % 5 + 1;
5      sleep(wait_time);
6     if (winner == 0) winner = id;
7      return NULL;
8  }
9
```

Concorrenza!

- Chi é avvantaggiato?
- Che problemi da la riga rossa?

TOR VERGATA

Side Quest Challenge 2

Side Quest 1

Assegnare un nome ad ogni thread e stampare il nome del thread vincitore e non il numero

Side Quest 2

Use malloc per gestire tutte le strutture necessarie

Side Quest 3

Aggiungere classifica da stampare alla fine da parte del main thread (ignorare ancora problemi di sincronizzazione).

TOR VERGATA

Challenge 3

Pseudo-grep 2: La vendetta

Scrivere un programma C multi-thread che prende tramite linea di comando una parola e N files.
Il programma deve creare N thread dove ognuno deve prendere in input un solo file e cercare il numero di volte in cui la parola e' stata trovata.

Side Quest Challenge 3

Side Quest 1

Il main thread deve aggregare il numero di ripetizioni trovate.

Side Quest 2

Ogni thread mostri a schermo non solo il numero ma anche la riga e del file in cui ha trovato la parola. (potete assumere che se la parola sia a cavallo tra due righe restituire la righa su cui inizia).

Esercizio Scheduling

Scheduling VRR

Si descriva lo scheduling di CPU round-robin virtuale.
Si consideri uno scenario in cui lo scheduler di CPU round-robin virtuale sia basato su un time-slice pari a $K > 5$ millisecondi, esistano due processi CPU-bound A e B di durata infinita ed esista un unico processo I/O-bound C di durata infinita. Il processo C ha CPU burst di lunghezza pari a 5 millisecondi.
Si indichi il massimo valore di K che possa permettere al processo B di avere un tempo di attesa per l'esecuzione del CPU burst non più ampio di 15 ms.

Round-Robin Virtuale (VRR)

Idea

- Variante del Round-Robin
- Due code:
  - coda principale
  - coda ausiliaria (processi da I/O)

Meccanismo

- Se un processo va in I/O prima di finire il quanto:
  - conserva il tempo residuo
  - al ritorno entra nella coda ausiliaria
- La coda ausiliaria ha **priorità**

Effetto

- Favorisce processi I/O-bound
- Migliora il tempo di risposta

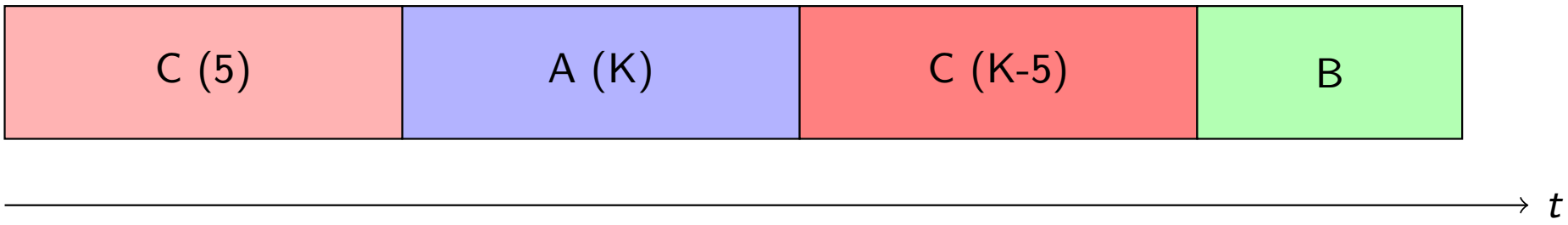
TOR VERGATA

Esercizio VRR – Esecuzione

Dati

- $K > 5$ ms
- A, B: CPU-bound C: I/O-bound (5 ms)

Timeline (caso peggiore per B)



Osservazione

- C usa $5 + (K - 5) = K$
- Attesa di B: $K + K = 2K$

Vincolo

$$2K \leq 15 \Rightarrow \boxed{K_{\max} = 7.5 \text{ ms}}$$

Esercizio Scheduling

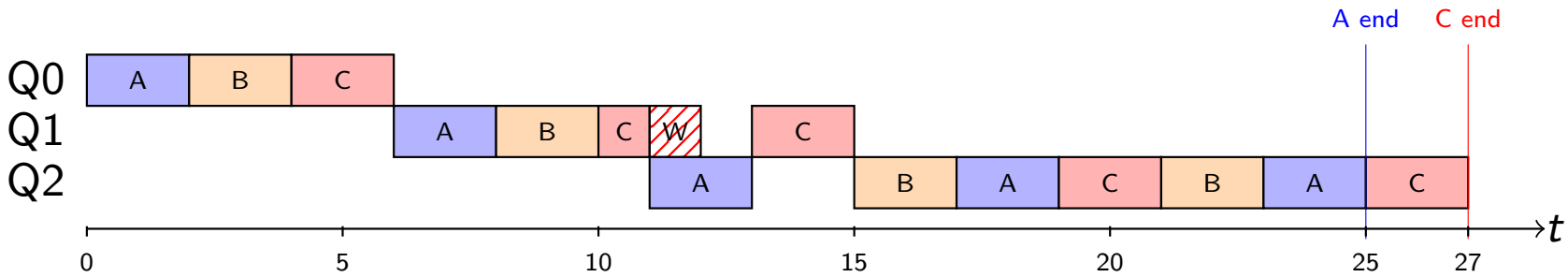
Scheduling MLFQ

Scheduler di CPU multi-level-feedback-queue. Supponendo che il quanto di tempo base di tale scheduler sia pari a 2 millisecondo e che al tempo $T = 0$ siano creati in ordine 3 processi A, B e C tali che A abbia un solo CPU-burst di durata pari a 10 millisecondi, B sia CPU-bound di durata infinita e C abbia due differenti CPU-burst di durata pari a 3 e 6 millisecondi, si determini il tempo di completamento di C considerando il caso in cui lo scheduler abbia 3 differenti livelli di priorità e tra i suoi due CPU-burst C rimanga in stato "wait" per 0.5 millisecondi.
Si assuma che la latenza per eseguire attività di sistema operativo sia trascurabile.

Assunzioni

- ogni livello a quanto di 2 ms
- A: 10ms
- B: inf
- C: 3 ms + 6 ms con I/O = 0.5 ms

Timeline (compatta)

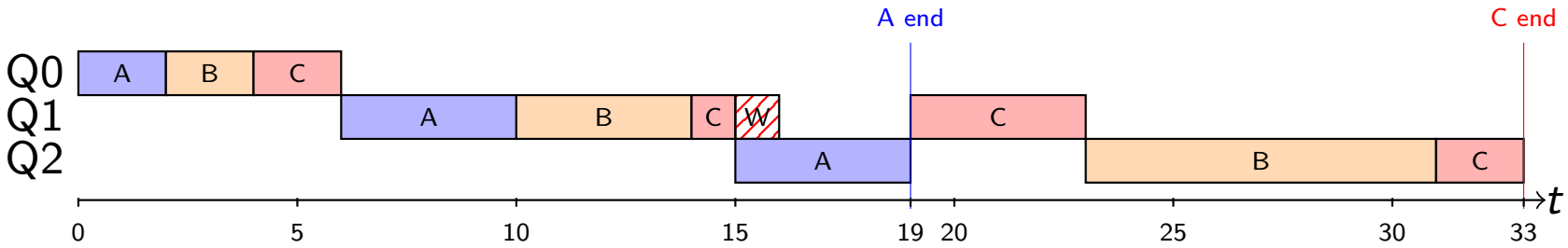


MLFQ – Timeline compatta

Assunzioni

- Esponenziale q0=2ms, q1=4ms, q2=8ms
- A: 10ms
- B: inf
- C: 3 ms + 6 ms con I/O = 0.5 ms

Timeline



Challenge 4

Raccolta di frutta

Scrivere un programma C che simula la raccolta della frutta.
Il main thread deve creare N thread, divisi in 3 gruppi, ognuno addetto alla raccolta di un tipo di frutto diverso: mele, pere ed arance.
Dopo aver creato i thread figli il main thread si mette a generare randomicamente i diversi tipi di frutta in una singola variabile condivisa con tutti i thread.
I thread leggono questa variabile condivisa e "raccolgono" la frutta a loro corrispondente aggiungendo il valore del frutto in una loro variabile personale.
Dopo un certo tempo il main thread smette di generare frutta, manda un messaggio di uscita nella variabile condivisa e stampa i valori di frutta raccolti Ignorare problemi di sincronizzazione.

Esercizio Scheduling

Scheduling VRR: Esercizio 1 Esame 13/6/22

Si descriva lo scheduling di CPU round-robin virtuale.
Si consideri uno scenario in cui lo scheduler di CPU round-robin virtuale sia basato su un time-slice pari a $K > 10$ millisecondi, esista un unico processo CPU-bound A di durata infinita ed esista un unico processo I/O-bound B di durata infinita. Il processo B ha CPU burst di lunghezza pari a 10 millisecondi.
Si indichi il massimo valore di K che possa permettere al processo B di avere un tempo di attesa per l'esecuzione del CPU burst non più ampio del 250% rispetto alla durata del timeslice.

TOR VERGATA