link testo esame

- Esercizio c1
- Esercizio c2
- Esercizio g1
- Esercizio g2

Esercizio c1

La fase finale di un torneo sportivo coinvolge 2^N squadre numerate $0, 1, ..., 2^N - 1$ e si svolge ad eliminazione diretta. Al primo turno la squadra 0 gioca con la squadra 1, la 2 con la 3 e così via. Al secondo turno la vincente fra le squadre 0 e 1 incontra la vincente fra 2, 3 ecc. Il procedimento continua fino ad individuare la sola squadra vincente del torneo. Scrivere il monitor torneo. L'attività delle squadre è la seguente:

```
Squadra: process( i,i {0, 1 ,..., 2^N-1 } )
    for turno in range(1, N):
        forma = valutaforma(i, turno)
        if torneo.gioca(i, turno, forma) == False:
            print(f"la squadra {i} ha perso al {turno} turno")
            return
    print(f"la squadra {i} ha vinto il torneo")
```

La procedure entry `gioca` di volta in volta confronta il valore del parametro `forma` delle due squadre e dichiara vincitrice quella con valore massimo, se i valori coincidono decide in modo casuale (rigori). Alla squadra vincente viene restituito il valore `True`, alla perdente il valore `False`.

```
#define SQUADRE 2^N  // numero di squadre che partecipano al torneo

class tree{
    int value = -1;
    tree parent = null;
    tree left = null;
    tree right = null;
    condition giocato;
};

monitor torneo {
    tree albero
    tree foglie[SQUADRE]
    condition

    procedure entry gioca(int squadra, int turno, int forma){
        if
    }
```

```
}
è assurdo passo.
```

Esercizio c2

Quella che segue vorrebbe essere l'implementazione di un semaforo unfair inizializzato a zero ma è errata. Mostrare un caso nel quale il valore del semaforo (value) non venga correttamente mantenuto. Correggere il codice usando il passaggio del testimone (passing le batôn).

```
class: wrongsem
int value = 0, count = 0
semaphore mutex init 1;
semaphore s init 0;
void wV():
 mutex.P()
 if value == 0 && count > 0:
    s.V()
 else:
    value++
 mutex.V()
void wP()
 mutex.P()
 if value == 0:
    count++
    mutex.V()
    s.P()
    mutex.P()
    count--
 else:
    value--
 mutex.V()
```

Esercizio g1

Sia dato l'algoritmo di rimpiazzamento modulo che data una memoria di NF frame , se avviene un page fault in corrispondenza dell'i-mo elemento della stringa di riferimenti modulo sceglie come pagina vittima quella che occupa il frame i % NF.

Es. se NF fosse 3, la prima pagina viene messa nel frame 1 perché $1 \% 3 = 1$, se l'undicesimo elemento della stringa causa un page fault, si elimina la pagina nel frame 2 ($11 \% 3 = 2$).

A) Considerare la stringa di riferimenti seguente e mostrare il funzionamento di modulo nei casi NF = 3 e NF = 4: 1 2 3 4 5 3 3 3 1 5

B) modulo è un algoritmo a stack?

Esercizio g2

rispondere alle seguenti domande (motivando opportunamente le risposte): - a) Perché il nome del file non è memorizzato all'interno dell'i-node nei file system tipo UNIX (e.g. ext2/3/4)? - b) Un bug di tipo buffer overflow consente ad un attaccante di spedire più dati di quelli che il buffer di ricezione può contenere, tracimando così nelle aree di memoria di altre variabili. Come è stato possibile in tanti casi che venisse spedito codice macchina e che il programma vittima dell'attacco lo eseguisse? - c) come fa l'allocazione gerarchica ad evitare che si possano verificare casi di deadlock? - d) Per rendere uno scheduler round-robin più pronto a servire i processi interattivi si decide di dimezzare la durata del time slice. Quali effetti collaterali può avere la scelta

Risposte

- a.
- b.
- c.
- d.