link testo d'esame

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

–

Risposte

Esercizio c1

In un porto con una sola banchina utilizzabile occorre caricare cereali sulle navi. I camion portano i cereali al porto. Una sola nave alla volta può essere attraccata al molo, un solo camion alla volta scarica i cereali nella nave.

Il codice eseguito da ogni nave è:

```
nave[i] process:
    porto.attracca(capacità)
    porto.salpa()
    ... naviga verso la destinazione
```

Il codice di ogni camion è:

```
camion[j] process:
    while (1):
        quantità = carica_cereali()
        porto.scarica(quantità)
```

- I camion fanno la spola dai depositi alla nave.
- La nave arriva vuota e può salpare solo se è stata completamente riempita [la somma delle quantità scaricate dai camion raggiunge la capacità indicata come parametro della funzione attracca].
- Se un camion può scaricare solo parzialmente il suo carico rimane in porto e aspetta di completare l'operazione con la prossima nave che attraccherà al molo.

Scrivere il monitor porto.

```
monitor porto:
    navi = 0            // quante navi sono al porto
    MAX = 0             // capacità max della nave [cereali max]
    cereali_nave = 0    // quanti cereali sono sulla nave
    waiting_cereali = 0 // cereali sul camion che aspetta la prossima nave
    condition camion_ok2scarica, nave_ok2salpa, ok2attracca

    procedure entry attracca(capacita):
```

```
        navi++                          // aggiungo la nave al porto
        if (navi > 1)               // Ho delle navi in attesa nel porto che non possono prende
            ok2attracca.wait()  // aspettano!
        MAX = capacita          // salvo quanti cereali servono alla nave
        if (waiting_cereali > 0 && navi > 0)    // Ho dei cereali in attesa di una nave e h
            cereali_nave += waiting_cereali     // aggiungo i cereali in attesa alla nave
            waiting_cereali = 0                 // azzero i cereali in attesa perchè sono s
            camion_ok2scarica.signal()          // il camion può riprendere a scaricare

    procedure entry salpa():
        if (cereali_nave != MAX) // la nave non è piena
            nave_ok2salpa.wait()
        navi--                  // tolgo una nave dal porto
        if (navi > 1)               // Ho delle navi che attendono di attraccare?
            ok2attracca.signal()    // do il segnale che può attraccare

    procedure entry scarica(quantita):
        cereali_nave += quantita    // carico i cereali sulla nave
        if (cereali_nave > MAX)
            waiting_cereali += cereali_nave-MAX     // salvo i cereali che non sono stati s
            cereali_nave = MAX          // tolgo i cereali che non ci stanno dalla nave
            camion_ok2scarica.wait()    // il camion deve attendere la prossima nave
        camion_ok2scarica.signal()  // il camion è vuoto e può andare e lasciare il posto a
        nave_ok2salpa.signal()      // la nave può partire
```

Esercizio c2

Dato un servizio di message passing asincrono implementare un servizio di message passing testardo che consegna solo i messaggi ricevuti due volte con il medesimo contenuto da qualsiasi mittente [i due messaggi uguali possono provenire da mittenti diversi]. Il servizio di message passing testardo prevede due funzioni:

```
void tsend(msg_t msg, pid_t dest)
msg_t trecv(void)
```

Il message passing testardo ha lo stesso potere espressivo del message passing asincrono?

```
//ho a disposizione:    asincrono => asend, arecv
//devo implementare:    testardo
list messages = [];     // lista di tutti i messaggi ricevuti
void tsend(msg_t msg, pid_t dest){
    asend(msg, dest);
}
// invia solo messaggi ricevuti 2 volte con lo stesso testo
msg_t trecv(void){
```

```
    while True{
        msg = arecv(ANY);
        if msg in messages{ // ho trovato il messaggio
            messages.delete(msg);
            return msg;
        }
        messages.add(msg);  // non l'ho trovato e lo aggiungo
    }
}
```

Sì i due metodi hanno lo stesso potere espressivo, infatti posso implementare l'uno con l'altro.

Esercizio g1

Fornire un elenco di incongruenze che possono venir rilevate dal fsck [file system check] applicato a file system di tipo UNIX [e.g. ext2, bffs]. Per ogni tipo di incongruenza indicare come viene riscontrata e con quali operazioni di fsck può ripristinare la coerenza.

1. **Link non validi**: un link simbolico che punta a un file che non esiste più. fsck può rilevare questa incongruenza e offrire opzioni per eliminare o correggere il link.
2. **File duplicati**: due o più file che hanno lo stesso nome e inode. fsck può rilevare questa incongruenza e offrire opzioni per rimuovere una delle copie o rinominare una di esse.
3. **File inode non validi**: un file che non ha un inode valido. fsck può rilevare questa incongruenza e offrire opzioni per recuperare il file o eliminarlo.
4. **Cartelle danneggiate**: una directory che contiene informazioni corrotte o che non può essere aperta. fsck può rilevare questa incongruenza e offrire opzioni per riparare la directory o eliminarla.
5. **Superblock danneggiato**: il superblocco, che contiene informazioni importanti sul file system, è corrotto o danneggiato. fsck può rilevare questa incongruenza e offrire opzioni per riparare o ricostruire il superblocco.

Esercizio g2

rispondere alle seguenti domande [motivando opportinamente le risposte]:

a. perchè un algoritmo di rimpiazzamento a stack non può essere soggetto di anomalia di Belady?

b. le password [criptate] nei sistemi UNIX moderni sono memorizzate in un file inaccessibile agli utenti [`/etc/shadow`]. Perchè?

c. nello scheduler di tipo SRTF [versione preemptive di STF], il tempo residuo può diventare negativo. Perchè?

d. quando si usano dischi a stato solido i file system vengono configurati in modo da non aggiornare il tempo di ultimo accesso ai file [noatime]. Perchè?

Risposte

- a. un algoritmo a stack non può essere soggetto ad anomalia di Belady in quanto dovrebbe avvenire page fault nel caso della mamoria con più frame.
- b. per evitare che edurante eventuali attacchi non siano vulnerabili e possano essere rubate. Questo file è però accessibile ad utenti amministratori del sistema.
- c. Nello scheduler di tipo SRTF il tempo residuo può diventare negativo per una sottostima del tempo di completamento del processo. Si verifica quando l'algoritmo sottostima la quantità di tempo necessaria per completare il processo corrente e lo scheduler passa ad un altro processo prima che il processo corrente sia effettivamente completato.
- d. Questo permette di migliorare le prestazioni, in quanto il tempo di ultimo accesso ai file con un SSD non è utile quanto con i dischi rigidi che ne necessitano per recuperare più velocemente le informazioni. Con gli SSD si cerca sempre di ottimizzare i cicli di scrittura in quanto non infiniti: la scrittura di questa informazione aumenterebbe il numero di cicli consumati, diminuendo la durata del disco.