Prova 2019-09-13

Esercizio c.1

```
monitor mbuf:
    coda buff<type>;
    coda buff_cond<int counter, condition freeable>
    int elem_counter;
    condition space

    void add(type data, int n):
        if (elem_counter == MAXELEM):
            space.wait()
        else:
            elem_counter++

        buff_cond.enqueue(n, new condition)
        for (i to n):
            buff.enqueue(data)


    type get(void):
        elem = buff_cond.get_first()
        #ritorna il riferimento al primo elemento della coda senza toglierlo da essa

        if (elem.counter > 1):
            elem.counter--
            elem.freeable.wait()
        else:
            buff_cond.dequeue()
            if (elem_counter == MAXELEM):
                space.signal()
            else:
                elem_counter--

        elem.freeable.signal()
        #anche se ne chiamo uno in più (il primo porcesso che voleva il dato) non è problema

        return buff.dequeue()
```

Esercizio c.2

```
process bohm[i, i=0,1] {
    for (;;) {
        m.pre(i);
        print(i);
```

```
        m.post(i);
    }
}

monitor m:
    condition ok[2];
    state = 0;

    procedure entry pre(int n) {
        if (state != n)
            ok[n].wait();
    }

    procedure entry post(int n) {
        state = 1 - state;
        ok[state].signal()
    }
}
```

Soluzione

```
semaforo ok[2] = {new semaforo(0), new semaoforo(0)}
semaforo ok_counter[2] = {0, 0}
//processi in coda sul semaforo
int state = 0

process bohs[i, i=0,1] {
    for (;;) {
        pre(i);
        print(i);
        post(i);
    }
}

void pre(int n) {
    if (state != n) {
        ok[n].P()
        ok_counter[n]++
}

void post(int n) {
    state = 1 - state
    if (counter[state] == 0)
        ok[state].V();
}
```

Esercizio g.1

Parte 1

1. Il processore esegue p1 per x millisecondi
2. p1 va in waiting mentre il device fa IO quindi viene eseguito p2 per x millisecondi
3. p2 lui va nella coda waiting e la sua richiesta per il device aspetta che quella di p1 sia terminata (se lo è già y minore x allora prende il controllo del device).
4. p3 viene eseguto per x millisecondi
5. p3 va nella coda waiting
6. vengono eseguiti in ordine p1, p2, p3 ciascuno per x millisecondi

Parte 2

I due scheduler si comportano allo stesso modo se: 1. $m >= x$ 2. $y > x$

La prima condizione ci garantisce di annullare la parte preemptive di round-robino, infatti ogni processo finirà l'esecuzione prima che l'interval timer cambi processo da eseguire.

La seconda condizione annulla la parte di prioritò del secondo scheduler. Infatti il primo processo che va in waiting andrà nella coda ready dopo y millisecondi. Se x > y allora P2 dovrebbere essere ancora in esecuzione quando P1 va in ready, per cui appena P2 va in waiting P1 torna in esecuzione. con y>x garantisco che quando P1 va in Ready P2 ha finito è già in waiting e sto eseguendo P3.

Esercizio g.2

a) Tramite la tecnologia degli interrupt il processore evita di fare busy waiting, stando ad aspettare che le operazioni di IO vengano terminate. In questo modo il processore può eseguire un altro processo mettendo in pausa quello che aveva fatto richieste IO, tornando a eseguirlo quando l'interrupt di termine viene lanciato.

b) Non avrebbe molto senso, infatti a prescindere dal modo in cui sono sparsi gli strip di parità, avremmoml'equivalente di un disco dati e uno di parità, ma quello di parità sarebbe semplicemente il bit dell'altro disco (non ci sono altri dischi con i quali calcolare tale parità).

c) Il paginatore entra in funzione quando avviene un pagefault. L'algortimo di rimpiazzamento invece quando non ci sono frame liberi e bisogna sbarazzarsi una pagina. La scelta della pagina vittimi è data da tale algoritmo.

d) I vantaggi sono che il programma occupa meno spazio in memoria. Inoltre ci possono essere aggiornamenti retroattivi, infatti basta aggiornare la libreria dinamica e tutti i processi che la sfruttano avranno funzioni aggiornate. Lo svantaggio è che si può aggiornare una libreria solo se è

retrocompatibile, e diversi programmi potrebbero implementare funzioni diverse.