link testo esame

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

Esercizio c1

Esercizio c2

Un servizio di message passing sincrono senza selezione del mittente prevede una API con due funzioni:

```
sasend(msg_t msg, pid dest);
msg_t sarecv(void);
```

La funzione sarecv restituisce il primo messaggio ricevuto da qualsiasi mittente ed è bloccante se non ci sono messaggi pendenti. la funzione sasend si blocca fino a quando il messaggio msg non viene ricevuto dal processo dest. Dato quindi un servizio di message passing sincrono senza selezione del mittente implementare un servizio di message passing sincrono (standard, quello definito nel corso) senza fare uso di processi server

```
waiting{
    dest;
    next;
}
void ssend(msg_t msg, pid_t dest){
    sasend(<msg, getpid()>, dest);
    waiting.append(dest);       // aggiungo nella lista il destinatario da cui aspettiamo A
    bool lock;
    waiting_ack = waiting;
    do {
        <ack, pid> = sarecv();
        do {
            if (pid == waiting_ack.dest){   // ho trovato il pid dell'ack ricevuto nella lis
                waiting.remove(waiting_ack.dest);   // lo rimuovo dalla lista perchè l'ho sb
            }
            waiting_ack = waiting_ack.next;
        } while (!waiting_ack.isempty())
        lock = waiting.isempty() ? false : true;    // ho sbloccato tutti? no=> devo aspett
    } while (lock)
}

lista {
    msg_t msg;
```

```
    pid_t pid;
    next;
};
msg_t srecv(pid_t mitt){
    bool check;
    do{
        <msg_t msg, pid_t pid> = sarecv();  // pid = chi ha mandato la recv
        if (mitt == pid){
            sasend(<ACK, getpid()>, pid);
            return msg;
        }
        // altrimenti cerco il mittente nella lista
        if (!lista.isempty()) {     // esiste la lista?
            while (lista.next != null) {
                if (lista.pid == mitt){
                    sasend(<ACK, getpid()>, lista.pid);
                    messaggio = lista.msg;
                    lista.remove(lista.pid, lista.msg);     // tolgo dalla lista il messagg
                    return messaggio;
                }
                lista = lista.next;
            }
        }
        lista.append(msg, pid);     // salvo nella lista i dati appena ricevuti
        check = true;   // devo controllare di nuovo in attesa di un nuovo messaggio
    }while (check)
}
```

Esercizio g1

Considerare i seguenti processi gestiti mediante uno scheduler preemptive a priorità statica su una macchina biprocessore SMP:

```
P1: cpu 4 ms; I-O 4 ms; cpu 2 ms
P2: cpu 2 ms; I-O 4 ms; cpu 5 ms
P3: cpu 5 ms; I-O 3 ms; cpu 3 ms
P4: cpu 10 ms; I-O 1 ms
```

l'Input-Output avviene su un'unica unità. Per il processo P1 ha priorità massima seguito da P2, P3 e P4 in sequenza decrescente, le richieste di I/O sono gestite in ordine FIFO. Calcolare il tempo necessario a completare l'esecuzione dei 4 processi.

risposta

22 ms

Esercizio g2

uguali a quelli del mese dopo