Prova pratica di Calcolatori Elettronici (nucleo v6.*)

C.d.L. in Ingegneria Informatica, Ordinamento DM 270

25 febbraio 2016

1. Due processi possono comunicare tramite una `pipe`, un canale con una estremità di scrittura e una di lettura attraverso il quale viaggia una sequenza di caratteri. I caratteri inviati dall'estremità di scrittura possono essere letti dall'estremità di lettura.

   Per realizzare le `pipe` aggiungiamo le seguenti primitive (abortiscono il processo in caso di errore):

   - `natl inipipe()` (tipo 0x5c, da realizzare): Crea una nuova pipe e ne restituisce l'identificatore (`0xFFFFFFFF` se non è stato possibile creare una nuova pipe).
   - `void writepipe(natl p, char *buf, natl n)` (tipo 0x5d, da realizzare): Invia `n` caratteri dal buffer `buf` sulla pipe di identificatore `p`. È un errore se la pipe `p` non esiste.
   - `void readpipe(natl p, char *buf, natl n)` (tipo 0x53, già realizzata): Riceve `n` caratteri dal dalla pipe di identificatore `p` e li scrive nel buffer `buf`. È un errore se la pipe `p` non esiste.

   Previdamo un tipo di `pipe` con buffer interno. La `writepipe`, trasferisce i byte nel buffer interno e la `readpipe` li preleva dal buffer. La `writepipe` blocca il processo chiamante solo quando il buffer è pieno e la `readpipe` solo quando il buffer è vuoto. Entrambe ritornano al chiamante solo quando l'intero trasferimento è stato completato (quindi è possibile che il processo sia bloccato e risvegliato più volte).

   Per semplicità non trattiamo i casi in cui più di un processo voglia accedere alla stessa estremità della stessa pipe.

   Per descrivere una `pipe` aggiungiamo al nucleo la seguente struttura dati:

```
struct des_pipe {
    natl not_full;
    bool writer_waiting;
    natl not_empty;
    bool reader_waiting;

    char buf[BUFSIZE];
    natl head;
    natl tail;
    natl n;
};
```

   Il campo `not_full` è l'indice di un semaforo di sincronizzazione su cui attendere che il buffer non sia pieno; Il booleano `writer_waiting` è vero se e solo se lo scrittore è bloccato sulla pipe; Il campo `not_empty` è l'indice di un semaforo di sincronizzazione su cui attendere che il buffer non sia vuoto; Il booleano `reader_waiting` è vero se e solo se il lettore è bloccato sulla pipe; I campi `buf`, `head`, `tail` e `n` servono a realizzare il buffer interno come una coda circolare (`n` è il numero di byte che il buffer contiene).

   Modificare i file `sistema.cpp` e `sistema.S` in modo da realizzare le primitive mancanti.

   SUGGERIMENTO: è possibile utilizzare le primitive semaforiche.