Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

8 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, da realizzare): Riceve n caratteri dalla pipe di identificatore p e li scrive nel buffer buf. È un errore se la pipe p non esiste.

Prevediamo un tipo di pipe senza un buffer interno. Per la writepipe, questo vuol dire che i caratteri possono essere trasferiti solo se un altro processo è pronto a riceverli tramite una readpipe, altrimenti la primitiva deve attendere. Inoltre, il processo che ha invocato la readpipe potrebbe aver chiesto meno caratteri di quelli da inviare: in questo caso si devono inviare i caratteri possibili e continuare ad attendere; questa operazione potrebbe ripetersi più volte fino a quando tutti i caratteri non sono stati trasferiti. Analoghe considerazioni valgono per la readpipe.

Per semplicità non trattiamo i casi in cui più di un processo voglia accedere alla stessa estremità della stessa pipe. Inoltre, assumiamo che i buffer passati alla **readpipe** e alla **writepipe** appartengano allo spazio utente comune.

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

```
struct des_pipe {
   natl reader_ready;
   natl write_done;
   char *r_buf;
   natl r_pending;
};
```

Il campo reader_ready è l'indice di un semaforo di sincronizzazione usato per notificare che un processo è in attesa di completare una readpipe, nel qual caso r_pending contiene il numero di caratteri ancora da trasferire e r_buf l'indirizzo a cui devono essere trasferiti. Il campo write_done è l'indice di un semaforo di sincronizzazione usato per notificare che il trasferimento richiesto dall'ultima readpipe è stato completato.

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

SUGGERIMENTO: è possibile utilizzare le primitive semaforiche.